Interactive Japanese-European text generation

- an approach to multilingual export translation based on Systemic Functional Grammar

A thesis submitted to the University of Manchester
for the degree of M.Sc. in the Faculty of Technology
1993

Graham Wilcock

Centre for Computational Linguistics
University of Manchester Institute of Science and Technology

Abstract

The translation quality of current machine translation systems is limited by restriction to syntactic and semantic factors. A wider range of factors, including pragmatic and discourse factors, needs to be extracted from the source text and transferred to the target text. However, correct generation of the target text often requires factors which are not present in the source text. One way to add these factors is by interactive inquiries to the user.

Systemic Functional Grammar (SFG) includes the necessary wider range of factors. It is also orientated explicitly towards the organization of choices in the values of these factors. The aim of the thesis is therefore to investigate the possible advantages in the use of SFG in machine translation. This investigation is in two parts: a survey of the theory and practice of SFG in both monolingual and multilingual generation, with reference to related issues in machine translation, and an implementation of a demonstration prototype showing an SFG-based approach to the inclusion of the wider range of factors by interactive inquiries.

The demonstration prototype is an interactive generation system, in which inquiries concerning factors needed in the generation of complex English modal and auxiliary verb groups are presented in Japanese to a Japanese user. The necessary factors are organized as an SFG network of choices, and the inquiries are based on felicity conditions for the appropriateness of the choices. The prototype system can be used for iterative revision of the generated sentences, under the control of the grammar. This suggests an approach to knowledge-based post-editing.

The thesis concludes with a brief comparison of the strong and weak points of SFG and HPSG, and considers the benefits if their strong points could be combined.

Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Acknowledgements

This work was done as a visiting researcher of Sharp Corporation at UMIST. I wish to thank Mr Mikio Ohsaki and Mr Shigeki Kuga of Sharp Corporation, and Prof Juan Sager and Prof Jun-ichi Tsujii of UMIST, for giving me this opportunity.

Many thanks to my supervisor, Bill Black, and to John Bateman, Brian Chandler, Robin Fawcett, Kristiina Jokinen, Takashi Kamiko, Osamu Nishida, John Phillips, Erich Steiner, Pete Whitelock, Kenji Yoshimura and others for their help and valuable discussions. To Pat Olver, and to Jiri and Susan Jelinek, I owe more than thanks.

Contents

Introduction

Aims of the thesis

Problems in machine translation

In his invited lecture on "Future directions of machine translation" at COLING-86, Jun-ichi Tsujii discussed the need for machine translation (MT) to take into account a wider range of factors in order to achieve improvement in translation quality.

> "...We have to extract explicitly more kinds of information from source texts than deep case structures and utilize these to compute descriptions of the target sentences." [Tsujii 1986]

He described a framework for future MT systems, in which source texts are analysed into "a set of monolingual factors which collectively determine surface structures of source texts", and target texts are generated from "a set of monolingual factors which collectively determine surface structures of target texts". These factors need to include not only syntactic and semantic factors, but also discourse factors, pragmatic factors, and further "factors of certain aspects of understanding". The transfer stage between analysis and generation would need to invoke understanding processes which are required for the specific language pair.

However, he also pointed out that generation of the target texts needs to use other information, which is frequently impossible to obtain from the source text.

> "It often happens that to determine target surface expressions requires a set of factors which are not expressed at all in the source language..." [Tsujii 1986]

Such factors occur particularly frequently in translation from Japanese to English and other European languages, as Japanese does not express definiteness or number of noun phrases, and regularly omits subjects and objects of verbs. I experienced direct confirmation of the difficulties caused by these missing factors, in my own work on English generation in Japanese-English MT systems (Sections 4.1.1 and 4.2.1).

An approach to the inclusion of the wider range of essential factors in the generation stage, following Tsujii's proposed framework, requires the following three problems to be solved:

1. The definition of a set of factors of various aspects of meaning (semantic, pragmatic, discourse...) which collectively determine the surface structure of target language texts,

2. The design and implementation of a method for generating the specific target language text determined by a given set of values of these factors,

3. The development of new methods for correctly deciding the values of these wider factors, including not only analysis of the source text but also other means, such as interactive user participation.

Solutions from Systemic Functional Grammar

Given these problems, Systemic Functional Grammar (SFG) has two particular attractions. First, unlike some other theories, SFG emphasises the wider range of factors (semantic, pragmatic, discourse...) discussed in [Tsujii 1986]. More specifically, semantic factors are handled in the ideational metafunction, pragmatic factors in the interpersonal metafunction, and discourse factors in the textual metafunction (see Section 1.2.3). Second, again unlike other theories, SFG is directly orientated towards the explicit organization of the functional choices needed to decide the values of these factors.

The general aim of this thesis is therefore to investigate SFG and its use in English and multilingual generation, and to consider its relevance to the problems of Japanese-English and Japanese-European MT. The more specific aim is to show that SFG offers an approach to solutions for all three of the above problems, as follows.

By traversing a systemic grammar network (Section 1.2.1) from left to right, using an "inquiry semantics" approach (Section 2.2.2) to the choices in each system, a set of values of various factors (semantic, pragmatic, discourse...) is selected. The realization rules (Section 1.2.2) for the selected

values then generate the structures and words of the output sentence. So a systemic grammar network defines a set of factors of various aspects of meaning which collectively determine the surface structure of target language texts. The realization rules define a method for generating the specific target language text determined by a given set of values of these factors. The inquiry semantics interface defines the inquiries which need to be raised to obtain all (and only) the information needed to decide the values of these factors. In relation to the three problems enumerated above:

1. The systemic grammar network provides a definition as in problem 1.

2. The realization rules provide a method as in problem 2.

3. The inquiry semantics interface provides an approach to problem 3.

This correspondence between the SFG approach to generation and Tsujii's proposed approach to MT was the motivation for the further investigation of SFG and for the development of the cross-linguistic demonstration prototype, which form the contents of this thesis.

Structure of the thesis

The thesis is divided into three parts: Part I gives a background survey of SFG and its application to natural language generation, and a review of some related issues in machine translation. Part II describes an implemented and working prototype system which demonstrates an SFG-based approach to the three problems listed above, which could be applied to target language generation in Japanese-English and Japanese-European MT. Part III briefly evaluates the prototype and the SFG-based approach, and suggests some areas in which an exchange of ideas between SFG and other approaches might be fruitful.

In Part I, Chapter 1 reviews the basic concepts of SFG on which the rest of the study is based. These include the organization of functional choices into a network, the way in which these choices are realized in surface structure, the coverage of the wider range of factors (semantic, pragmatic and discourse factors), and the way in which the different factors are combined in generation.

Chapter 2 describes two large computational implementations of SFG applied to English generation: the USC Penman system and the Cardiff COMMUNAL system. The descriptions include the grammars

on which the systems are based, the ways in which inquiries for functional choices are made, and the other components of the generation environment in which the grammars are embedded.

Chapter 3 looks at current research on the problems involved in extending SFG-based generation from monolingual English to multilingual generation. Two further projects are described: the Sydney project on SFG-based generation of English, Chinese and Japanese, and the Darmstadt KOMET project on generation of German, English and Dutch. The descriptions include the functional approach to multilingual grammar sharing, the representation of multilingual grammars, and problems in handling grammatical gender agreement in SFG.

Chapter 4 reviews some basic issues in machine translation (MT), including the important differences between import and export translation, and the need to reduce the extreme dependence on skilled post-editors. Some research approaches towards the idea of MT for monolingual users are described. Multilingual MT and multilingual generation are compared, as two alternative solutions to the problems of international organizations. The survey ends with ideas for an SFG-based approach to MT.

In Part II, the demonstration prototype is described. Chapter 5 gives details of the small grammar embedded in the prototype. The organization of the choice networks of the grammar is explained.

Chapter 6 describes the new implementation of the grammar to meet the requirements of the demonstration prototype. The method of representation for the grammar network and the realization rules is shown, and the implementation of the procedures is described, with extracts from key parts of the code.

Chapter 7 explains the design of the Japanese interface, and its use for interactive English generation. Illustrative examples of interactive sentence revision are presented in the form of window images.

Chapter 8 describes the extension of the prototype to include French and German generation. The grammar network is shared by the three languages, but separate realization rules were developed. Examples of multilingual sentence revision using the Japanese interface are presented.

In Part III, Chapter 9 evaluates the prototype. The possibility of applying this approach to knowledge-based post-editing is considered.

Chapter 10 looks at strengths and weaknesses of SFG. These are briefly contrasted with corresponding weaknesses and strengths of HPSG. The thesis concludes with a brief examination of the idea of combining the strengths of the two theories, and the potential benefits of doing so.

Part I

Background survey

Chapter 1

Systemic Functional Grammar

1.1 Origins of SFG

Systemic Functional Grammar (SFG) is a major linguistic theory, which has received special attention from researchers working in natural language generation. It was developed at the University of London by Michael Halliday, as a continuation of the work of his predecessors there, in particular that of J.R. Firth. This approach is therefore sometimes called "Neo-Firthian linguistics", or the "London school of linguistics" [Sampson 1980] (though both Firth and Halliday came from Leeds). [Steiner 1983] gives an excellent survey of the development of the theory. Here only a few key points in the growth of SFG will be mentioned.

Firth's predecessor at London, the anthropologist B.K. Malinowski made important contributions to early modern linguistics from an anthropological perspective. His view of "meaning as function in

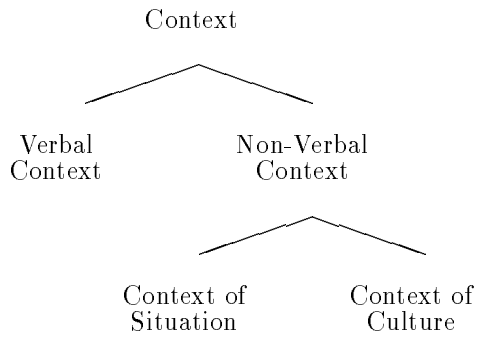

Figure 1.1: Malinowski's analysis of context

context" was inherited by Firth and Halliday. His analysis of different types of context, summarised in Figure 1.1 (from [Steiner 1983]), was the forerunner of Halliday's division of the functional areas of language into three general metafunctions (see Section 1.2.3).

J.R. Firth, the founder of modern British linguistics and the first Professor of General Linguistics in the UK, continued Malinowski's emphasis on a social and functional approach to language, while establishing linguistics as an independent discipline. It was Firth who began to use the word "system" in a new sense as a technical term, from which the name "systemic grammar" originated.

> "The first principle of analysis is to distinguish between STRUCTURE and SYSTEM. Structure consists of elements in interior syntagmatic relation and these elements have their places in an order of mutual expectancy. ... Systems of commutable terms or units are set up to state the paradigmatic values of the elements." [Firth 1957]

Firth emphasized the need for linguistics to give equal importance to both the "anatomy" and "physiology" of language. These two aspects of language can be summarized in the list of contrasting pairs of Firthian linguistic terms in Figure 1.2.

"anatomy"	"physiology"
chain	choice
syntagmatic	paradigmatic
structural	systemic
formal	functional
logical	rhetorical

Figure 1.2: The "anatomy" and "physiology" of language

Firth disagreed with the American structuralists of his time (led by Bloomfield), because they were concerned only with the "anatomy" of language. For the same reason Michael Halliday, Firth's pupil and successor at London, disagreed with the American formalists (led by Chomsky). Halliday is closer to the European functionalists, such as the Prague school, from whose theory of Functional Sentence Perspective he adopted the theme/rheme structure for his textual metafunction (see Section 1.2.3). In reaction to the dominant American schools, the Neo-Firthians stressed the "physiology" of language, not because the "anatomy" is unimportant, but in an attempt to redress the balance.

Whereas Firth's theory was often expressed in general terms, and his concrete examples were often fragmentary, Halliday developed a systematic and comprehensive theory of language, with a new terminology of its own. This theory, expounded in Halliday's many publications, became known as Systemic Functional Grammar. It was called "systemic" because of his development of detailed system networks

(see Section 1.2.1) for many areas of English grammar, and for interesting areas of other languages. It was called "functional" because of his development of the theory of the ideational, interpersonal and textual metafunctions (see Section 1.2.3).

One unusual aspect of Halliday's theory is his non-acceptance of morphology as a separate level of language. He showed how inflections could be handled by systems and realizations in exactly the same way as clause structures. His approach here may have been influenced by the relatively restricted morphology in the two languages - English and Chinese - in which he specialised.

1.2 Some basic concepts of mainstream SFG

There is no single comprehensive and authoritative textbook of SFG. [Halliday 1985] is authoritative but explicitly only covers the functional part of SFG, not the systemic part. A number of Halliday's system networks are given in Chapter 9 of [Kress 1976].

The comprehensive system networks of the USC Nigel/Penman project (see Section 2.2) are based directly on Halliday's publications and his personal participation in the project, but only fragments have been published. The examples in this section are drawn from this project, specifically from [Matthiessen 1991].

1.2.1 System networks

A system consists of an *entry condition* and a set of *output features*. There is no upper limit to the number of output features, but most systems have only two. The basic form of a system is shown in Figure 1.3. This system has the system name "MOOD TYPE", a single entry condition "clause", and two alternative output features "indicative" and "imperative". The system represents the paradigmatic

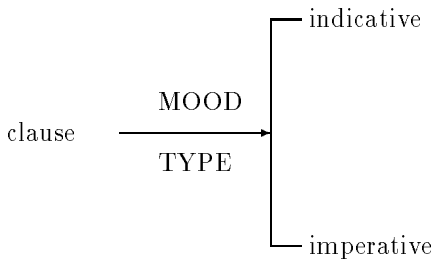

Figure 1.3: A basic system diagram

choice between these features, for major clauses in English.

An output feature of one system may be the entry condition for another system. In this way, systems are linked together into a system network. A fragment of a system network for the English mood system is shown in Figure 1.4. This figure contains three separate systems: MOOD TYPE, INDICATIVE TYPE and INTERROGATIVE TYPE. The output feature "indicative" of the MOOD TYPE system is also the entry condition of the INDICATIVE TYPE system, so that there is a dependency between these two systems: the INDICATIVE TYPE system is only entered if "indicative" is chosen as the output feature of the MOOD TYPE system.

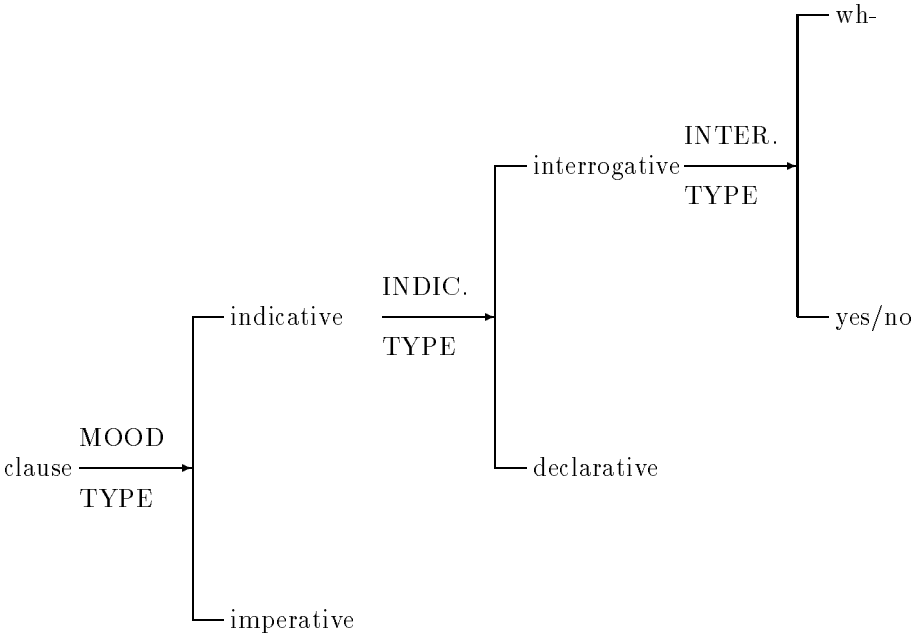

Figure 1.4: A fragment of a system network diagram

In SFG, the dependencies between systems in the network also has a more general significance. The more fundamental systems occur towards the left of the network, and the more "delicate" systems occur towards the right of the network. This left-to-right dimension of the network is called the *scale of delicacy*. [Mellish 1988] discusses this aspect of system networks in terms of *incremental description refinement*.

More than one system may share the same entry condition. In this case, the systems are entered in parallel from the entry condition. In the system network diagram a left curly bracket { is drawn from the entry condition, spanning all the simultaneously entered systems. An example is shown in Figure 1.5.

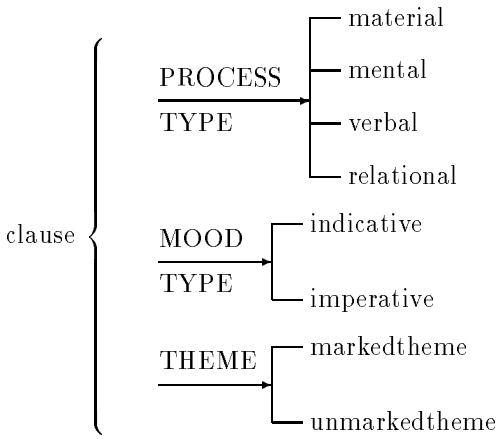

Figure 1.5: Simultaneous systems

A system may have more than one entry condition. If two or more entry conditions are conjointly *necessary* for entry to the system, they are linked to it by a right curly bracket }. If any one of two or more entry conditions is disjointly *sufficient* for entry to the system, they are linked to it by a right square bracket].

Systems represent paradigmatic choices not only between grammatical alternatives, but also between lexical alternatives. The lexicon is considered as a thesaurus. There is no clear division between grammar and lexicon, and Halliday uses the term *lexicogrammar* to include both. In general, grammatical choices occur towards the left of the network, and lexical choices occur towards the right of the network. This has given rise to the expression *lexis as most delicate grammar*.

The explicit description of paradigmatic choices is what distinguishes SFG from other approaches to grammar. Halliday described the choices which are available in a language as the "meaning potential" of that language. The systems show *meanings*, which are realized in the structures of the language as *wordings* (see Section 1.2.2).

1.2.2 Realization

Realization rules show how the paradigmatic choices in the systems are expressed as syntagmatic chains in the structures of the language. The process of realization can be considered as a mapping from the "physiology" to the "anatomy" of Figure 1.2, i.e. from choices to chains, from function to form.

In mainstream SFG, realization rules are embedded in the system network with the output features

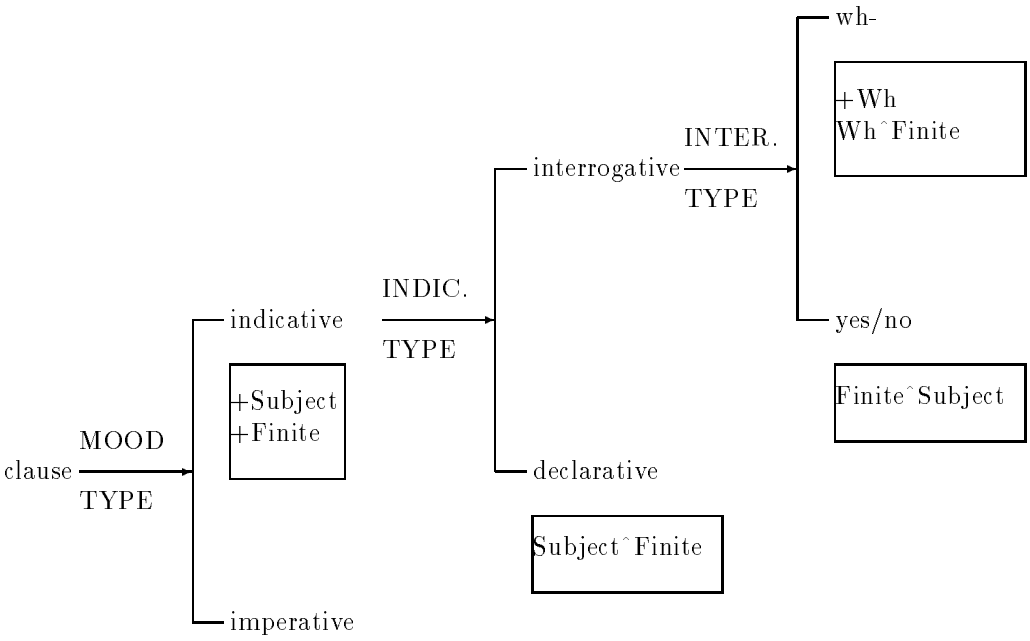

Figure 1.6: System network diagram with realization rules

which they realize. Figure 1.6 shows the system network fragment of Figure 1.4 with realization rules added. Here, the output feature "indicative" has two associated realization rules: "+Subject" and "+Finite". These are both *insertion* realizations. The output feature "declarative" has one associated realization rule: "Subject^Finite". This is an *ordering* realization.

There are slight variations within SFG in the definitions, the terminology and the notations used for realization rules. [Berry 1977] gives a clear explanation of the types of realization and the stages of the realization process. She carefully distinguishes the *inclusion of functions* from the realization of the functions by the *insertion of elements of structure*. But in mainstream SFG including the Nigel grammar used here for illustration, these two stages are merged, giving the following main types of realization rules.

- *Insertion* of a grammatical function. In Figure 1.6 "+Subject" requires the presence of a subject when a clause is indicative, and "+Finite" requires the presence of a finite verb when a clause is indicative.

- *Ordering* of grammatical functions. In Figure 1.6 "Subject^Finite" requires the subject to precede the finite verb when an indicative clause is declarative.

- *Conflation* of grammatical functions. Figure 1.6 only shows a fragment of a system network from the interpersonal metafunction. Other parts of the network will insert grammatical functions

from other metafunctions (see Section 1.2.3), such as "+Agent" in the ideational metafunction, and "+Theme" in the textual metafunction. The realization of these different aspects of meaning in a single wording is achieved by conflation of the functions (see Section 1.2.4). For example "Agent/Subject" conflates the Agent and the Subject, and requires that they will be realized by the same element of structure.

- *Preselection* of features. Realization is effected at several successive levels, with a partial traversal of the system network for each level. For example after traversal of the clause part of the network a Subject may have been inserted, and a traversal of the noun group part of the network will then realize the internal structure of the Subject. Realization rules at the clause level may be used to preselect certain features, to be obligatory during traversal at the noun group level.

- *Exponence* of an element. This type of rule requires that an element of structure is realized by a specific lexical item.

1.2.3 Metafunctions

Halliday developed a theory of the fundamental functions of language, in which he analysed lexicogrammar into three broad metafunctions: ideational, interpersonal and textual. Each of the three metafunctions is about a different aspect of the world, and is concerned with a different mode of meaning of clauses. The ideational metafunction is about the natural world in the broadest sense, including our own consciousness, and is concerned with clauses as *representations*. The interpersonal metafunction is about the social world, especially the relationship between speaker and hearer, and is concerned with clauses as *exchanges*. The textual metafunction is about the verbal world, especially the flow of information in a text, and is concerned with clauses as *messages*. Malinowski's influence (see Figure 1.1) seems clear here: the ideational metafunction relates to the context of culture, the interpersonal metafunction relates to the context of situation, and the textual metafunction relates to the verbal context.

In each metafunction an analysis of a clause gives a different kind of structure composed from a different set of elements. In the ideational metafunction, a clause is analysed into *Process, Participants* and *Circumstances*, with different participant types for different process types (as in Case Grammar). In the interpersonal metafunction, a clause is analysed into *Mood* and *Residue*, with the mood element

further analysed into *Subject* and *Finite*. In the textual metafunction, a clause is analysed into *Theme* and *Rheme* (as in the Prague School).

<table>
<tr><td>In this job</td><td>Anne</td><td>we</td><td>'re</td><td>working</td><td>with silver</td><td></td></tr>
<tr><td colspan="2">Theme</td><td colspan="4">Rheme</td><td>textual</td></tr>
<tr><td rowspan="2"></td><td rowspan="2">Vocative</td><td colspan="2">Mood</td><td colspan="2" rowspan="2"></td><td rowspan="2">interpersonal</td></tr>
<tr><td>Subject</td><td>Finite</td></tr>
<tr><td>Locative</td><td></td><td>Actor</td><td colspan="2">Process</td><td>Manner</td><td>ideational</td></tr>
</table>

Figure 1.7: Metafunctional layering

Figure 1.7, taken from [Matthiessen & Bateman 1991], shows an analysis of the sentence "In this job, Anne, we're working with silver" into three different structures in the three metafunctions. This kind of diagram is called a "metafunctional layering" diagram in SFG, but the metafunctions do not have any kind of relative "depth", rather they are different dimensions.

The metafunctional theory is part of the "functional" side of SFG, but it is also important in the "systemic" side of SFG. Each metafunction has a principal system in the networks for clauses, verbal groups and nominal groups. For example the TRANSITIVITY system is the principal system for the ideational metafunction in the clause network. These principal systems are shown in Figure 1.8, taken from [Matthiessen & Bateman 1991].

	ideational	interpersonal	textual
clause	TRANSITIVITY	MOOD	THEME
verbal group	TENSE	MODALITY	VOICE
nominal group	MODIFICATION	PERSON	DETERMINATION

Figure 1.8: Principal systems

An important theoretical point is that in general, in the system networks, the systems *within* each metafunction are closely interconnected, but are largely independent of systems in the other metafunctions. System interconnections *across* metafunctions are rare. This is illustrated in Figure 1.9, taken from [Matthiessen & Halliday to appear].

In this network fragment, there are normal dependency relationships within the MOOD region of the interpersonal metafunction, between the MOOD-TYPE and INDICATIVE-TYPE systems and between the INDICATIVE-TYPE and INTERROGATIVE-TYPE systems, and there is also a further interconnection: the TAGGING system can be entered either from the imperative feature of the MOOD-TYPE

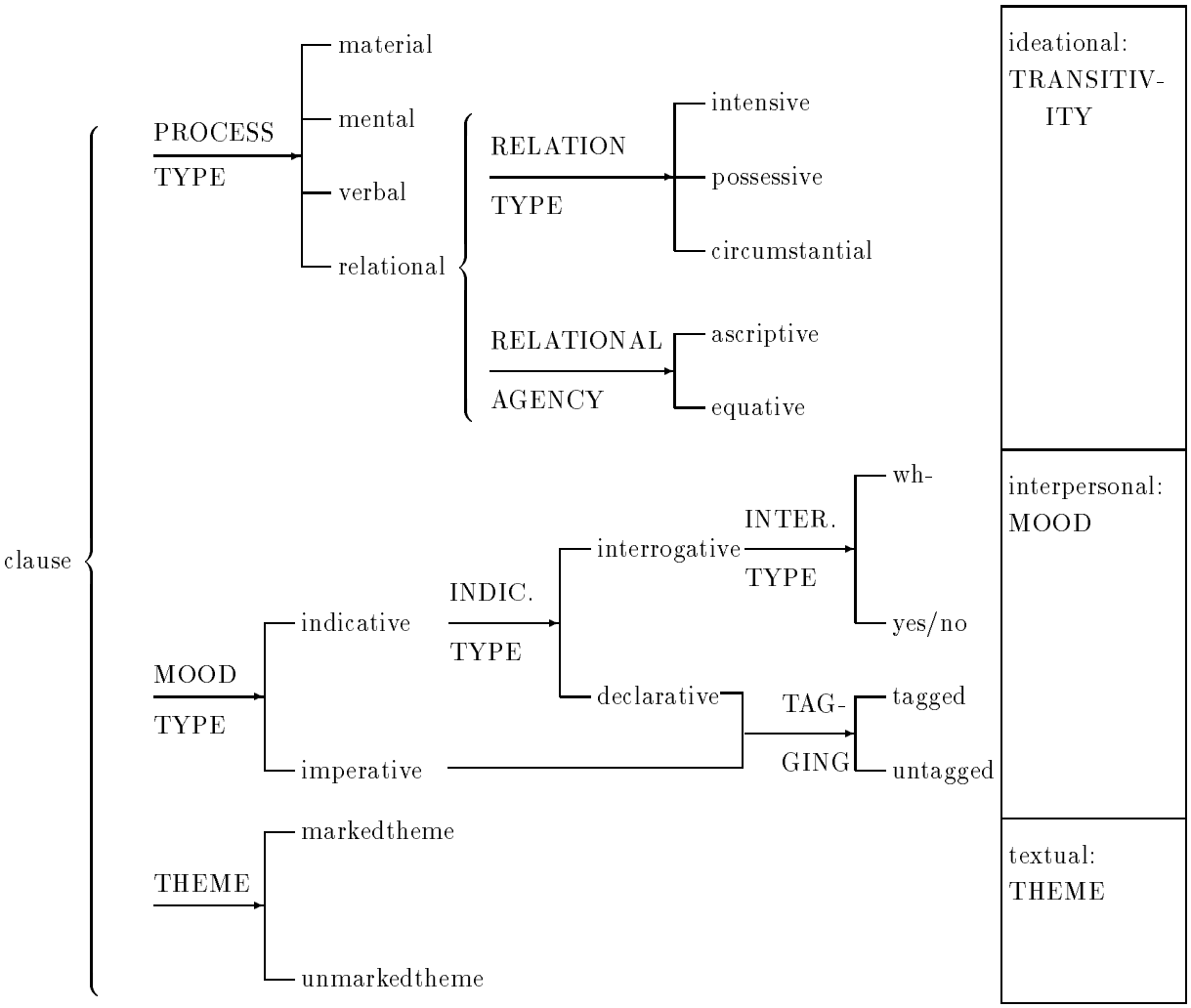

Figure 1.9: Independence of metafunctions

system or from the declarative feature of the INDICATIVE-TYPE system. But there are no interconnections at all between the MOOD region of the interpersonal metafunction and the TRANSITIVITY region of the ideational metafunction.

1.2.4 Conflation

Although there is a high degree of independence between the three metafunctions on the level of paradigmatic choice systems, the process of realization combines the paradigmatic choices from all three metafunctions into a single realization on the level of syntagmatic chain structure. This process of merging in realization is performed by a specific realization operation called *conflation*.

The conflation operation is preceeded by *insertion* operations, which insert grammatical functions as a step in the realization of paradigmatic choices in the three metafunctions. For example, certain choices

in the systems of the ideational metafunction are realized by the insertion of the grammatical function Actor, other choices in the systems of the interpersonal metafunction are realized by the insertion of the grammatical function Subject, and other choices in the systems of the textual metafunction are realized by the insertion of the grammatical function Theme. The conflation operation then combines these grammatical functions in various ways, specified by the realization rules for further choices made in the interpersonal and textual metafunctions. This is illustrated in Figure 1.10 (taken from [Halliday 1985]).

(a)

I	caught the first ball
Theme Subject Actor	

(b)

I	was beaten by	the second
Theme Subject		Actor

(c)

the third	I	stopped
Theme	Subject Actor	

(d)

by	the fourth	I	was knocked out
	Theme Actor	Subject	

Figure 1.10: Different conflations

In Figure 1.10 example (a), all three grammatical functions are conflated into a single element of structure: Subject, Actor and Theme are all realized by the same element "I". This is the unmarked case in English, an active declarative clause. In example (b), the two grammatical functions Subject and Theme are conflated, and again realized by the element "I", but Actor is not conflated with them in this passive clause. In example (c), Subject and Actor are conflated, but not together with Theme. Although the clause is active, a marked theme has been chosen in the textual metafunction. In example (d), Theme and Actor are conflated, but not with Subject. The clause is passive like (b), but this time the Actor is thematized.

1.3 Textual coherence and cohesion

One of the strong areas of SFG research is in aspects of language which concern complete texts as opposed to single sentences. In contrast to the formalist approach, which views the sentence as the basic unit in linguistic analysis, SFG does not consider the sentence to be a particularly important unit. In SFG, the clause is the most important unit in grammatical analysis, but clauses are sub-units of texts. Texts have textual features, functions and structures of their own, which affect the clauses and clause complexes which occur within them.

Texts are analysed in SFG in terms of *field*, *tenor* and *mode*, corresponding respectively to the ideational, interpersonal and textual dimensions of the metafunctional theory. Different text types are analysed in SFG work on *genre*, including studies of *register variation*. The typical structures of different text types are described in terms of their *generic structure potential*.

Unfortunately, although SFG has a lot to say about the many issues involved in textual structure, coherence and cohesion [Halliday & Hasan 1989], these issues lie outside the scope of this thesis. The demonstration prototype, which is concerned with cross-linguistic user interaction in multilingual generation, generates only single clauses. Therefore text-level issues will only be mentioned briefly in this background survey, despite the fact that for some of the projects described these issues are of fundamental importance.

Chapter 2

Text generation

2.1 Generation with SFG

2.1.1 Parsing-oriented versus generation-oriented grammars

Most of the grammatical theories and formalisms used in computational linguistics (such as LFG, GPSG, UCG and HPSG) have been developed as part of the intensive work on natural language analysis and parsing over the past two decades, based on a formal rather than functional approach to grammar. The parsing-oriented perspective and the form-oriented approach of these formalisms go together naturally. The starting point for parsing is a given input string, and the task is to produce from the surface forms of the string an appropriate analysis into syntactic structures and semantic contents using a grammar. The starting point for their theorizing is that a language is a set of strings, and a grammar is a set of rules which specify those strings and the structures which are to be derived from them by analysis. The grammar formalisms therefore typically emphasise the surface properties of the strings. For example, emphasising word order, linear precedence information is (or was) typically included in phrase structure rules, and concatenation is typically the only string operation.

More recently, some of these formalisms (such as UCG and HPSG) have been applied seriously to generation. The emphasis has been on the importance of a declarative approach to grammar development, so that the grammar can be used not only for parsing but also "backwards" for generation. One of the problems in generation with these approaches is that the logical form used as the starting point

for generation is typically under-specified in interpersonal and textual information. This is discussed further in Sections 10.2.1 and 10.2.2.

By contrast, SFG has been applied mainly in the development of text generation systems, and is based on a functional approach to grammar. This generation-oriented perspective and function-oriented approach also go together naturally. The perspective of SFG is that the starting point for generation is a functional grammar representing a language as a network of choices, and the task is to produce from it an appropriate synthesis into a structure and hence a string (a "wording"). The starting point for theorizing is that a language has a set of functional choices, and a grammar is a network of systems which specify those choices, plus realization rules which produce the appropriate wordings. The set of functional choices of a language is called by Halliday the "meaning potential" of the language.

SFG-based systems, especially the USC Nigel/Penman system (see Section 2.2), have made a major contribution to the development of the field of natural language generation in the past decade, but only a few rather experimental attempts have been made to use SFG for analysis. These are surveyed in [O'Donnell 1992]. Two interesting approaches to running SFG "backwards" for parsing, emphasising a declarative approach to grammar development, are based on the representation of SFG in the FUG and TFS formalisms. This is discussed further in Section 10.2.3.

2.1.2 Structure-driven versus grammar-driven generation algorithms

In text generation systems, a distinction is usually made between *strategic generation* (what to say) and *tactical generation* (how to say it). Many systems divide these functions into two separate modules, a text planner and a surface generator. The text planner is executed first, and the resulting text plan is then passed to the surface generator for realization. The interface is usually some version of a logical form representation of semantic content.

This separation gives the usual benefits of modularization, breaking down the overall generation problem into smaller parts, but it creates a double-sided problem of how to match the linguistic coverage of the two components. One side of this problem is how to guarantee that the surface generator includes the linguistic resources necessary to realize the given text plan, and does not simply fail. The other side of the problem is what to do when the surface generator's grammar generates more than one output string from a given logical form. Both sides of the problem arise because the generation algorithm is

driven by the structure of the logical form, and searches the grammar for rules which match the current substructure. If there is no matching rule, the generator fails, and if there is more than one matching rule, the generator produces more than one output (or somehow has to select one of the rules).

In SFG-based systems, the generation algorithm is *driven by the grammar, organized as a network*, not by a logical form. The grammar network is traversed, the output feature from each system deciding which system or systems must be entered next. Therefore exactly one output will always be produced. The range of possible outputs corresponds to the "meaning potential" of the given grammar.

The problem in the SFG approach is therefore a different one: how to make the choices in all the systems. The choices have to be the "correct" ones, to express the content of the text plan, and to convey the correct communicative intent. The Penman project approached this problem by developing the *inquiry semantics* or *chooser/inquiry interface* (see Section 2.2.2). The COMMUNAL project initially intended to use *felicity conditions*, but has recently developed *predetermination rules* as a basis for the choices (see Section 2.3.2). My demonstration prototype simply presents the choices interactively to the user, but does this cross-linguistically, with brief canned-text paraphrases in the user's language, which is different from the languages being generated (see Section 7.2).

The choices have to come from somewhere outside the surface generator (the systemic grammar plus the traversal algorithm), from the environment in which it is used. The design of SFG-based generation systems has in fact followed the approach of building the lexicogrammar first and then using its demands for choices to *infer* how its environment should be organized to enable the correct choices to be made. This approach to the *design* of generation systems is described in [Matthiessen 1987].

2.2 The USC Nigel/Penman system

The Nigel/Penman system is the principal example of a fully-developed SFG-based generation system. It was developed at the University of Southern California Information Sciences Institute (USC/ISI) during the 1980's under the leadership of Bill Mann, with major contributions by Christian Matthiessen, Sandra Thompson, John Bateman, Robert Kasper, Eduard Hovy and others. Halliday's personal participation in the development of the Nigel systemic grammar, around which the Penman generation system was built, made this the "mainstream" SFG implementation.

As well as being possibly the biggest and most significant English generation system so far implemented, the Penman system has been the basis for other important developments in NLG. Mann, Thompson and Matthiessen's work on Rhetorical Structure Theory (see Section 2.2.4) at USC/ISI grew out of their work on the Penman project. Following Bateman's move to Darmstadt and Matthiessen's move to Sydney, their collaborating projects used Penman as the starting point for development of SFG-based generation systems for languages other than English, and for research on grammar-sharing and parallel multilingual generation (see Section 3.1).

2.2.1 The Nigel grammar

The heart of the Penman generation system is the Nigel systemic grammar of English [Matthiessen 1988]. It contains over 600 systems, starting with the RANK system which selects from the rank scale: clause, group, word or morpheme. However, Nigel is mainly a grammar of the clause and the group. At the bottom of the rank scale, at the word and morpheme levels, Nigel does not have the unified lexicogrammar of pure SFG theory (*lexis as most delicate grammar* - see Section 1.2.1). Instead there is a separate lexicon, which also includes a morphological system. At the top of the rank scale, at the clause level, Nigel covers *clause complexes* of two clauses in paratactic or hypotactic relation, but higher-level and more complex structures are handled outside Nigel by the RST system (see Section 2.2.4).

The systems in Nigel's grammar network take the form described in Section 1.2.1. During generation, the network is traversed iteratively and serially, proceeding generally down the rank scale (but with *rank shift* for clauses embedded in groups). When the output of a system satisfies the entry condition of two or more systems, they are added to a list of waiting systems, from which one is selected at random. The various operations in Nigel's realization rules are described in Section 1.2.2. Realization rules attached to an output feature are executed immediately the feature is chosen, except ordering rules which are collected and executed later.

The Penman system is implemented in Common Lisp, and the Nigel grammar is written in a Lisp-like notation. The grammar fragment shown earlier in Figure 1.6 is written as follows. (This example is adapted from [Teich 1992b] to match Figure 1.6. It includes *choosers* which are described in Section 2.2.2 below. The network diagram for this fragment, with the choosers incorporated, is shown in Figure 2.1 below).

```
(system
   :name MOOD-TYPE
   :inputs CLAUSE
   :outputs ((INDICATIVE (INSERT SUBJECT) (INSERT FINITE))
             (IMPERATIVE))
   :chooser MOOD-TYPE-CHOOSER
   :region MOOD
   :metafunction INTERPERSONAL)

(system
   :name INDICATIVE-TYPE
   :inputs INDICATIVE
   :outputs ((DECLARATIVE (ORDER SUBJECT FINITE))
             (INTERROGATIVE))
   :chooser INDICATIVE-TYPE-CHOOSER
   :region MOOD
   :metafunction INTERPERSONAL)

(system
   :name INTERROGATIVE-TYPE
   :inputs INTERROGATIVE
   :outputs ((YES/NO (ORDER FINITE SUBJECT))
             (WH (INSERT WH) (ORDER WH FINITE)))
   :chooser INTERROGATIVE-TYPE-CHOOSER
   :region MOOD
   :metafunction INTERPERSONAL)
```

2.2.2 The chooser/inquiry interface

To generate with a systemic grammar, a choice must be made in each disjunctive choice system during grammar network traversal. In SFG theory, these choices are made implicitly by speakers in order to express their communicative intent, but for a computational implementation an explicit formalization of the choice mechanism is necessary. For Penman, Mann developed the *inquiry semantics* or *chooser/inquiry interface* [Mann 1983].

Each choice system has an associated procedure called its *chooser*, which traverses a decision tree from its root to a single leaf node. The leaf nodes of the tree are the possible choices, and the geometry of the tree is arranged so that the branching nodes are simple binary decisions. Each branching node has an associated *inquiry*, which obtains information from the external environment in which the grammar is embedded. The chooser then selects which branch to take according to the reponse to the inquiry (see Figure 2.1, from [Matthiessen & Bateman 1991]).

Figure 2.1 shows the contrast between the grammatical terminology ("indicative"/"imperative") used in the labels on the system networks and the functional terminology ("command?") used in the

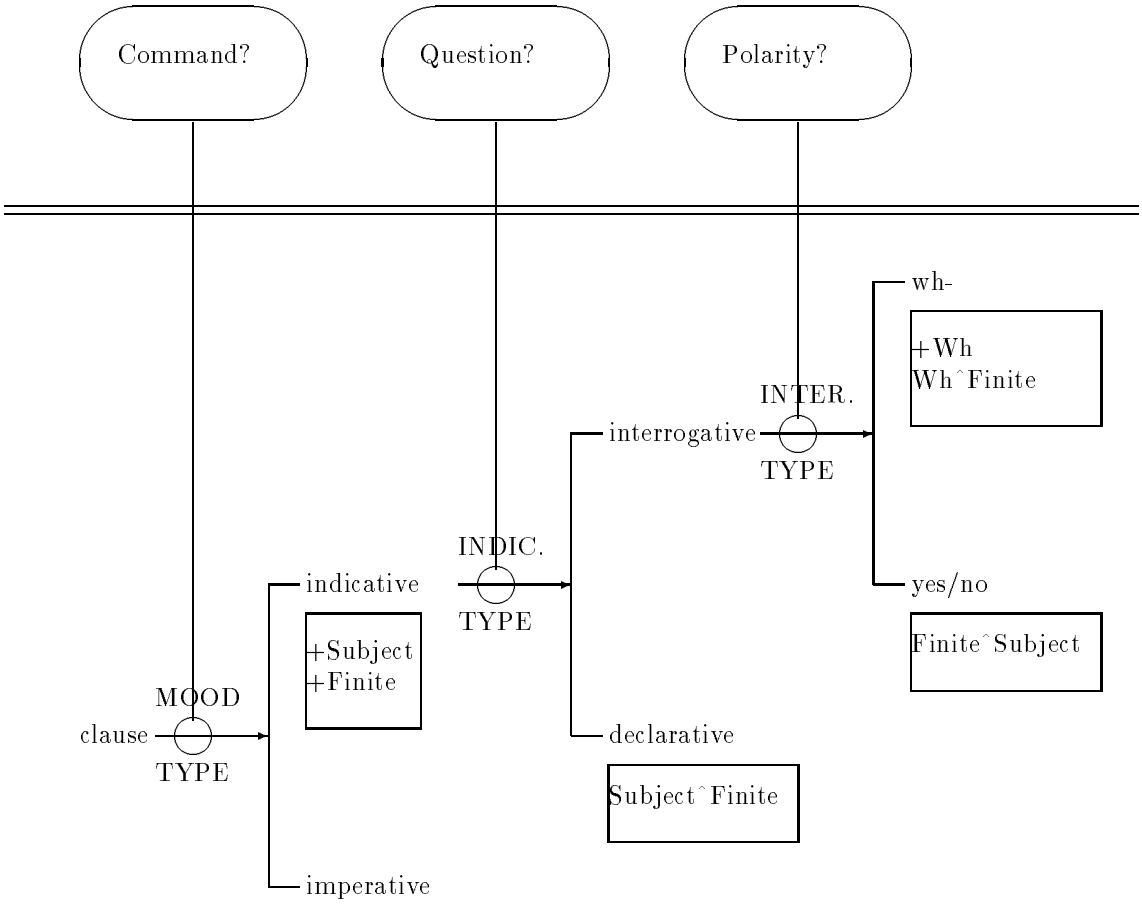

Figure 2.1: System network diagram with choosers and inquiries

inquiries raised by the choosers. This contrast is discussed further in the description of the COMMUNAL project in Section 2.3.1.

The applicability of the inquiry semantics approach to multilingual generation and export machine translation is discussed later in Section 4.4.

2.2.3 Upper Model and SPL

In the Penman generation system, the Nigel grammar is embedded in an environment which includes an *Upper Model*. An Upper Model is a particular domain model developed inside a general taxonomy, which is itself based on an ideational grammatical semantic typology for English known as the "Bloomington Lattice". Whereas most knowledge bases in AI applications are claimed to hold non-linguistic "propositional" knowledge, organized according to some apparently non-linguistic principles (logical or psychological, physical or metaphysical), the Upper Model in Penman reflects English lexical semantics (in fact, English lexicogrammatical semantics). But since it is a domain model, it reflects the *ideational*

metafunction (see Section 1.2.3). To distinguish it from the AI type of knowledge base, and because the interpersonal and textual metafunctions are usually not included, the Upper Model is called the *ideation base*.

Penman can be run interactively, in which case the inquiries are presented to the user as short natural language queries. But usually it forms part of some text generation application, in which case the interface between the application and the generator is the Sentence Plan Language (SPL). The SPL input includes not only ideational content but also interpersonal and textual specifications. The ideational content refers to the Upper Model, mainly by mapping referents from the ideation base onto ideational grammatical functions (such as Actor). The interpersonal and textual content specifies appropriate responses to the inquiries which will be raised by the generator.

A simplified example of an SPL expression, from [Matthiessen & Bateman 1991], is:

```
(p1 / class-ascription
       :domain      (A2 / adder
                        :identifiability-q identifiable)
       :range       (B1 / binary-operator
                        :identifiability-q notidentifiable))
```

Here Domain and Range are ideational grammatical functions, mapped to "adder" and "binary-operator" from the Upper Model. Identifiability-Q is the inquiry raised by the chooser of the IDENTIFIABILITY system, which has the two alternative output features identifiable and notidentifiable. Given a full version of this SPL input (with specifications needed for choice of tense, mood, etc.) Penman will generate the sentence "The adder is a binary operator".

2.2.4 Rhetorical Structure Theory

The highest-level *rank* in the Nigel grammar is that of clause-complex, which handles the relationship between main and subordinate clauses or between co-ordinated clauses. The work of Thompson, Mann and Matthiessen on higher-level text structures led to the development at USC/ISI of Rhetorical Structure Theory (RST). Originally RST was developed as a *descriptive* theory for use in the analysis of texts [Mann & Thompson 1988]. Subsequently Hovy developed a *constructive* version of RST for use in text generation [Hovy 1988].

Although developed as part of the work of the Penman project, RST is not part of SFG theory, which has other approaches to text structure [Halliday & Hasan 1989]. Recently text generation researchers

using various approaches have adopted RST, without necessarily adopting SFG. Some researchers have combined SFG and RST approaches, for example using system networks to build RST paragraph trees [Vander Linden *et al.* 1992]. Unfortunately, these areas lies outside the scope of this thesis.

2.3 The Cardiff COMMUNAL system

The COMMUNAL English generation system has been developed at the University of Wales College of Cardiff by a team led by Robin Fawcett. Unlike most other current SFG-based generation systems, it is not directly based on the Nigel/Penman system. An early version of Fawcett's approach is fully presented in [Fawcett 1980].

2.3.1 The Genesys grammar

The heart of the COMMUNAL system is the Genesys systemic grammar of English. Genesys contains about 600 grammatical systems, and is therefore of broadly similar size to Nigel. All other systemic generators (apart from those which are directly based on Nigel) are much smaller.

A small fragment of Genesys, extracted and modified to be free-standing as a coherent and fairly complete (within its stated coverage) mini-grammar for English modal and auxiliary verbs, was published as an appendix to [Fawcett 1988]. This mini-grammar, which is described in Chapter 5, was used as the basis of the demonstration prototype system for interactive Japanese-European generation described in Part II.

Functional level of the system network

Some systemic linguists, particularly Hudson in his earlier publications such as [Hudson 1971], concentrated more on the syntactic structures of language and less on the social and functional aspects, in an approach which was influenced by the dominant American formalist school. System networks were used primarily as a representation for the classification of syntactic structures and the variations between them. In this approach, the mapping between the systemic choices and their syntactic realizations was more direct, and the labels on the systems used a more syntax-oriented terminology.

Fawcett, by contrast, has concentrated more on the cognitive and social functions of language

[Fawcett 1980]. He uses system networks primarily as a representation of these functions, at an independent, functional level. This requires a more complex mapping from the systemic choices at this functional level to their realizations at the level of form. In this approach, the labels on the systems use a more function-oriented terminology.

Mainstream SFG, following Halliday, considers the system network as a representation of semantic and functional choices, which are often related fairly directly to alternative syntactic structures. Where the mapping is direct, syntax-oriented labels are used, and where it is not, function-oriented labels are used.

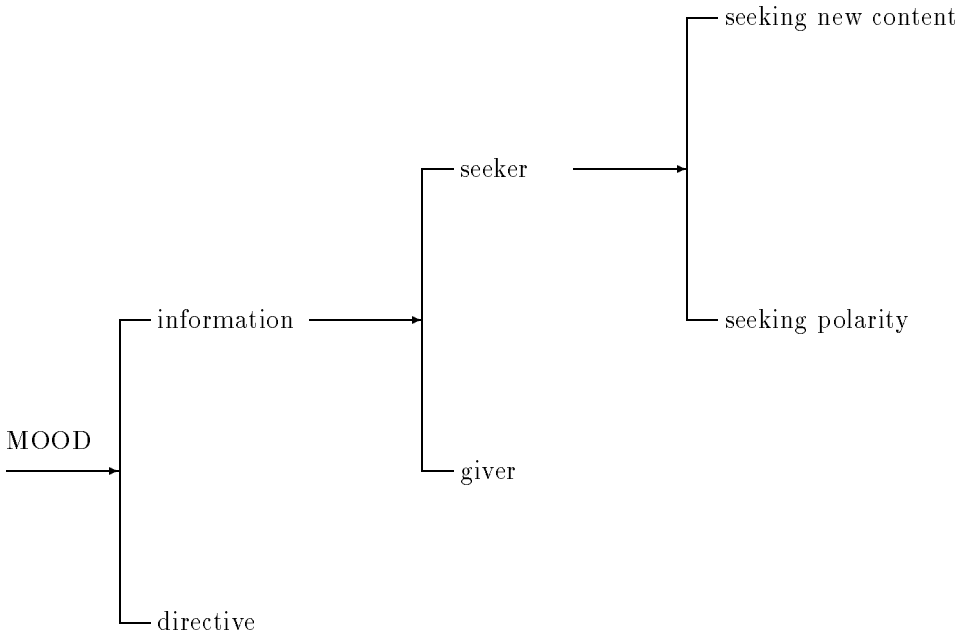

Figure 2.2: System network diagram with functional labels

An example of the more functionally-oriented labelling used on system networks in the Genesys grammar is shown in Figure 2.2, adapted from [Tucker 1989]. This shows the same choice systems as Figure 2.1 from the Nigel grammar. In Genesys, the alternatives are labelled "information"/"directive", rather than "indicative"/"imperative", (information-)"giver"/(information-)"seeker", rather than "declarative"/"interrogative", "seeking polarity"/"seeking new content", rather than "yes/no"/"wh-".

Comparison of these fragments shows the similarity between the functionally-oriented labelling of the system network in Genesys and the functionally-oriented terminology of the inquiries in Penman ("Command?", "Question?", "Polarity?"). Whereas in Penman the inquiries at the functional level are separate from the Nigel system network at the grammatical level, in COMMUNAL the Genesys system

network is itself the representation of the choices at the functional level.

The functional labelling adopted for the systemic choices in the Genesys grammar was helpful when this approach was applied to the cross-linguistic interface in the demonstration prototype, as described in Section 7.1.

Separation of realization rules

In Genesys the realization rules are not embedded in the system network with the output features which they realize. They are more complex than in Nigel, and are organized separately. The reason for the greater complexity is the functional level of the Nigel system network: a more complex mapping is required from this functional level to the level of form than is required from the grammatical level in Nigel.

The greater complexity of this mapping takes the form of complex *conditions* in the Genesys realization rules. The rules are similar to the production rules used in expert systems, with a left-hand side (LHS) stating a set of conditions and a right-hand side (RHS) stating a set of actions to be taken only if the set of conditions is true. The LHS consists of *conditional features*, each of which is "true" only if the named feature has been selected as an output feature in the traversal of the system network. The conditional features are combined into complex conditions by the Boolean operators NOT, AND and OR.

The features selected during a traversal of the system network are collected into a *selection expression*. The process of realization takes place after the network traversal, and consists of checking the truth of the LHS complex condition of each realization rule against the set of features in the selection expression, and executing the RHS set of realization operations if the complex condition is true. The realization operations are basically similar to those of Nigel, described in Section 1.2.2, and can include *re-entry* operations which cause a further traversal of the system network to fill substructures within the structure being realized.

The separation of the realization rules from the system network, adopted in Genesys for monolingual English generation, was helpful when this approach was applied to multilingual generation in the demonstration prototype, as described in Section 8.1.

Starting structures

Although the realization operations in Genesys are basically similar to those in Nigel, there is a difference in the approach to word order. In Nigel, the *ordering* operation specifies that two grammatical functions will be realized as adjacent elements of structure, one preceeding and the other following. The overall ordering of a complete series of elements forming a larger structure is worked out from the composition of these pair-wise orderings. In Genesys, an array of numbered slots called a *starting structure* is predefined for the larger structure, and the ordering operation for an element specifies which slot it is to fill.

In the very restricted mini-grammar used for the demonstration prototype, the starting structure for Clause has only 9 slots, as described in Section 5.2. In the full Genesys grammar, the starting structure for Clause has over 100 slots. This use of Fortran-like arrays with numbered positions seems inappropriate in natural language processing, where the use of recursive operations on data structures linked by pointers has been so successful in handling both potentially infinite and potentially empty structures elegantly. However, the SFG approach to constituency uses *ranked constituent analysis* of relatively flat and wide structures, as described in [Halliday 1985] Section 2.2, rather than the immediate constituent analysis of the dominant American formalist approach, so the use of fixed starting structures is not totally inappropriate.

Some of the slots in starting structures are marked as the default positions for specific elements. When an element is to fill its *unmarked* position, the explicit ordering operation can be omitted, only the insertion operation is necessary. In this way, starting structures hold some of the information which is explicitly stated in linear precedence rules in other approaches.

Probabilities and preselections

The output features of the choice systems in Genesys have probability values. The probability values are used to weight the choices in random generation, which can be started either from initial entry or from a point where some choices have already been made, to check or demonstrate the behaviour of the system.

The Nigel grammar includes preselection as one of its realization operations. During a traversal of the system network at a given rank (such as clause rank), the realization rules for a grammatical function at that rank may request a subsequent traversal of the network at another rank (such as group

rank) to select the features for the grammatical function and realize its structure. The realization rules for the grammatical function (such as Actor) may *preselect* some of the features (such as noun group, animate) to be selected in the subsequent traversal.

The Genesys grammar has a more sophisticated approach to preselection. When a realization rule requests a re-entry to the network to "fill" a grammatical function, the probability values to be used during the re-entry may be set to appropriate new values. Appropriate sets of probability values are defined in *preferences subrules* for specific grammatical functions, such as Agent. An example of the subrule for Agents is given in [Fawcett 1992b] as follows:

```
"Ag_preferences_subrule":
      for "Ag" prefer [thing,
            99% count / 1% mass,
            99% concrete / 1% abstract,
            99% living / 1% non_living,
            99% creature / 1% plant,
            98% human / 2% non-human,
            99% whole_human / 1% part_human],
      for "Ag" re_enter_at thing.
```

The absence of a probability value for the feature *thing* at the beginning of the list indicates that this feature is preferred absolutely, i.e. its probability value is to be 100%. The assignment of absolute preference to a feature is equivalent to preselection in Nigel. The assignment of relative preferences is similar to Wilksian preference semantics. A feature may also be inhibited for a given re-entry to the network, by setting its probability value to zero, while still permitting a choice to be made among the other output features in the choice system containing the inhibited feature.

Implementation and grammar representation

The COMMUNAL project is implemented in POPLOG Prolog. In contrast to the Lisp-like representation used for Nigel, the Genesys grammar is written in a Prolog-like notation. The system network fragment shown in Figure 2.2 is written as follows, adapted from [Tucker 1989]:

```
MOOD -> 90% information / 10% directive.
information -> 30% seeker / 70% giver.
seeker -> 50% seeking_new_content / 50% seeking_polarity.
```

2.3.2 Felicity conditions

As described in Section 2.1.2, the key problem in the SFG-based approach to generation is how to make the choices in all the choice systems. The systems specify the choices in terms of what the alternatives are, and the network specifies whether and when a choice is required in terms of the dependencies between systems. But the actual decision as to which alternative is the correct one has to come from somewhere outside the system network.

In Nigel, as described in Section 2.2.2, the approach to this problem is to have a decision tree for each system in its chooser, and to raise a functionally-oriented inquiry for each branch in the chooser. The response to the inquiry can come from another component of the overall Penman system, or it can be specified in advance in the input Sentence Plan Language (which may be the output from another component), or it can come from a human user when the system is used interactively. Many of the inquiries have a brief canned-text natural language form which can be displayed to clarify the choice to a human user.

In Genesys, as decribed in Section 2.3.1, the approach to this problem is to have functionally-oriented systems, so that the system network itself is a decision tree. It was, however, also originally intended that every choice would be given an associated *felicity condition*, which could be used by other components of the overall COMMUNAL system to decide on the correct choice. Many systems were given felicity conditions, in brief natural language form, which can be displayed to clarify the choice to a human user when the system is used interactively.

Although the chooser/inquiry interface in Nigel and the use of felicity conditions in Genesys are not identical, they both include the possibility of using brief canned-text natural language to enable a human user to make functional choices interactively. This approach was adopted in the demonstration prototype described in Part II, with the difference that the felicity conditions are displayed in a different natural language (Japanese) from the language generated by the system, as described in Section 7.1.

Recently, the COMMUNAL project has investigated the use of complex algorithms which apply weightings to factors from different components of COMMUNAL in order to *predetermine* the systemic choices. This recent work is described in [Fawcett 1992a].

2.3.3 Dialogue Model and other components

COMMUNAL differs from Penman in being orientated more specifically towards dialogue, while Penman is orientated more towards the generation of written texts. In SFG this is a distinction of *mode*, as mentioned briefly in Section 1.3. In Genesys the MODE system has two output choices: spoken or written. In spoken mode, the realizations include prosodic features and intonation patterns. The COMMUNAL approach to written texts is in fact to consider them as "extended monologues" in written mode.

The Genesys grammar is embedded in the overall COMMUNAL system together with other components, just as Nigel is embedded in the overall Penman system. However, the difference in orientation is reflected in the overall system. Penman's Upper Model is primarily an ideation base, and the interpersonal and textual factors are added explicitly in SPL statements, as described in Section 2.2.3. COMMUNAL, on the other hand, includes a Belief System as an ideation base, an Addressee Model for interpersonal factors, and a Discourse Model for textual factors.

The Discourse Model combines Rhetorical Structure Theory with a Systemic Flowchart Model of dialogue structure. This is described in [Fawcett 1992c], but lies outside the scope of this thesis.

Chapter 3

Multilingual generation

3.1 The importance of multilingual generation

Multilingual aspects of natural language processing have been the subject of much research in the field of machine translation, and have also been addressed in the field of database interfaces, but most work in the field of text generation has been restricted to monolingual systems, almost all for English.

However, there are important practical reasons for developing systems which can generate equivalent texts in several languages in parallel. International organizations like the EC and the UN need to produce large quantities of equivalent administrative documents on a regular basis in a fixed set of official languages. International companies need to produce equivalent product documentation in various sets of languages, especially for unified but multilingual market areas such as the EC.

These requirements are currently addressed by large-scale translation operations, but it is questionable whether translation is the right approach for these requirements. This question is discussed in Section 4.3. When all the documentation needs to be newly created in all the required languages, and when all the languages have equal status (so that there is no original document in one language which is "primary" or more "authentic" than the others), it seems more appropriate to generate all the texts at the same time.

In the case of the drafting of administrative documents, the source of information may be the recorded minutes of many separate committee meetings or legislative sessions, which may have taken place in various different languages which happened to be convenient for the meetings. Documents based on this

information then need to be drafted in all the official languages of the organization, following established standards and conventions for the document type. In the case of product documentation, the source of information may be a technical design database created during product development as part of a Computer-Aided Design process. A range of documents then needs to be produced, for widely-differing purposes (installation instructions, user guides, reference manuals, marketing literature, etc.), in the languages of the intended markets.

3.2 The Sydney multilingual SFG project

A group at the University of Sydney, led by Christian Matthiessen, are currently developing an SFG-based approach to multilingual generation. They have deliberately chosen to work with three languages - English, Chinese and Japanese - from three different language families, in order to establish a sound theoretical basis and methodology for the research, without taking advantage of the structural similarities between related languages. Following the functional approach of SFG, the project is based on identifying *functional* commonality across languages, rather than structural commonality. The research includes work on *grammar sharing* in SFG.

3.2.1 Grammar sharing

Other researchers, approaching grammar sharing from a unification-based perspective, have investigated techniques for sharing *syntagmatic, syntactic and structural* descriptions of different languages. For example, [Kameyama 1988] describes a process of "grammatical atomization" based on generalization of feature structures and unification of grammatical templates composed by multiple inheritance in an inheritance lattice. The focus of this research is "the syntactic well-formedness of nominal expressions" and "grammatical agreement, and word order types" [Kameyama 1988].

The Sydney research, by contrast, approaching grammar sharing from a functional perspective, is investigating techniques for sharing *paradigmatic, semantic and systemic* descriptions of different languages. The focus is "common functionality based on text in communicative context ... manifested in the paradigmatic, strategic organization of the resources of (a) stratum in the first instance and only secondarily in the tactic organization of the structural realization" [Bateman *et al.* 1991b].

3.2.2 Multilingual system networks

An example of the representation of common functionality in shared system networks is given in Figure 3.1 (adapted from [Bateman *et al.* 1991b]). This shows a fragment of the MOOD systems for English and Chinese. It differs from the monolingual fragment for English given in Figure 1.4 by the use of double lines for the MOOD-TYPE, INDICATIVE-TYPE and INTERROGATIVE-TYPE systems, to show that they are shared by both English and Chinese, and by the addition of a box containing two further systems, POLARITY-TYPE and TAG-TYPE, which are required for Chinese but not for English.

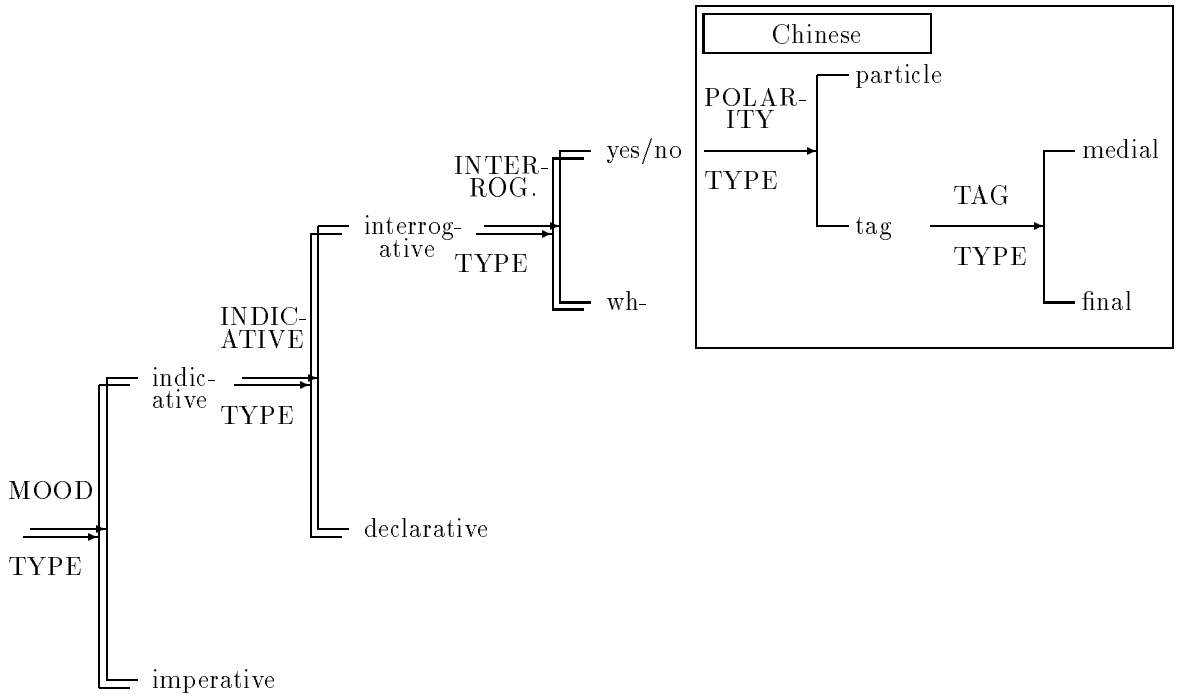

Figure 3.1: A fragment of a shared system network for English and Chinese

When Japanese is added, the shared network fragment is represented as in Figure 3.2 (adapted from [Bateman *et al.* 1991b]). The triple lines show that English, Chinese and Japanese share three common systems. The double lines and the language identifiers in the box around POLARITY-TYPE show that this system is shared by Chinese and Japanese but not English. The single lines and single identifier show that the TAG-TYPE system is specific to Chinese. A new system, PROJECTION-TYPE, which is specific to Japanese, is added in a box with a Japanese-only identifier. In Japanese, *other-projected* information, which a speaker has heard from another source, is realized by the *sou desu* form, in distinction to *speaker-projected* information, when the speaker is the original source, which is

realized by the unmarked form.

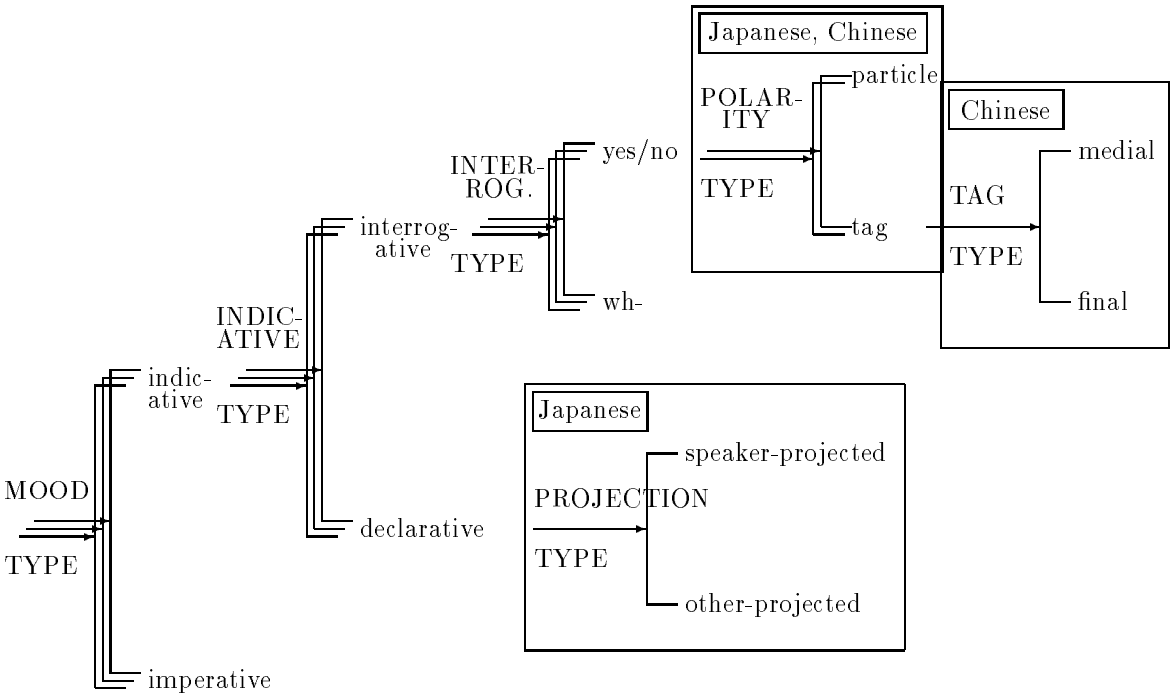

Figure 3.2: A fragment of a shared system network for English, Chinese and Japanese

3.2.3 Multilingual realization rules

The diagrams do not include realization statements. In the monolingual Nigel grammar, realization statements are normally shown in the network diagrams alongside the features which they realize, as in Figure 1.6. However, in the diagrams for multilingual networks, this practice has been dropped. Although such different languages as English, Chinese and Japanese share common functionality extensively, the structural realizations are typically quite different. Therefore the realization rules are set out separately for human readability (as in the COMMUNAL project).

In the computational implementation, the realization rules are still embedded inside the system definitions. As the realizations are generally different for the different languages, language identifiers need to be added to most of the realization statements. As the languages also differ in which systems are applicable, language identifiers also need to be added to many system definitions. The grammar representation therefore becomes quite complex. This is illustrated by one small example from [Bateman *et al.* 1991b]:

```
(system
   :name (:chinese :japanese POLARITY-TYPE)
   :inputs polarity
   :outputs
    ( (tag
         :chinese  (insert Tag)
         :japanese ((expand Process Positive Negative)... ))
      (particle
         :chinese  (lexify Interrogator ma)
         :japanese (order-at-end Interrogator)) )
   :chooser POLARITY-TYPE-CHOOSER)
```

3.3 The Darmstadt KOMET system

A group at Gesellschaft für Mathematik und Datenverarbeitung (GMD)/Integrated Publication and Information Systems Institute (IPSI) in Darmstadt, led by John Bateman, are working on the KOMET system for generation of German and multilingual text. Although Penman was the main starting point for development of the KOMET system, the German grammar, documented in [Teich 1992b], is strongly influenced by Fawcett's work. This is the result of close collaboration with the Eurotra-D group at Saarbrücken, led by Erich Steiner, who used Fawcett's approach to semantic relations in their work on German in the Eurotra and CAT2 frameworks, as described in [Steiner *et al.* 1988].

3.3.1 Grammatical gender agreement

The KOMET work on SFG-based generation of German has investigated areas which have been relatively neglected in previous computational implementations of SFG. Unlike English, Chinese and Japanese, German has both an extensive morphological system and extensive syntactic agreement requirements within noun groups. Morphology and agreement are based on number, gender and case. Gender in German is "grammatical" gender based idiosyncratically on individual lexemes, not "natural" gender based on animacy and sex as in English.

In the monolingual English generation systems described in Chapter 2, both number and gender have been handled as *semantic* features. The lexicogrammatical generation component (the systemic grammar) raises inquiries about these features, and the response comes from the ideation base, as described in Section 2.2.2. In German, grammatical gender is a *lexical* feature. For example, *der Wald* (the forest) is masculine. However, it makes no sense to ask the ideation base to specify the sex of a

forest.

The basis of the solution adopted in KOMET is to create a new pseudo-metafunction AGREEMENT, to include a LEXICAL-GENDER choice system in the lexicogrammar, and to arrange that the response to this system's LEXICAL-GENDER-CHOOSER comes from the lexicon, not from the ideation base. The system, from [Teich 1992b], is as follows:

```
(system
   :name LEXICAL-GENDER
   :inputs (OR
             NOMINAL-TERM-RESOLUTION
             RELATIVE-WHICH
             RELATIVE-THAT)
   :outputs ((MASCULINE-GENDER (CLASSIFY THING G-MASC))
             (FEMININE-GENDER (CLASSIFY THING G-FEM))
             (NEUTER-GENDER (CLASSIFY THING G-NEUT)))
   :chooser LEXICAL-GENDER-CHOOSER
   :region DEICTIC
   :metafunction AGREEMENT)
```

Given the grammatical gender of the head noun, obtained by the above system, another system DEICTIC-LEXICAL-GENDER is necessary to achieve syntactic agreement between the determiner and the head noun in noun groups, and another system is also needed to achieve agreement between adjectives and the head noun. The details are given in [Teich 1992b].

The KOMET researchers, faced with these diffficulties in implementing grammatical gender agreement in the SFG framework, have begun investigating the question of whether the notion of linguistic *dependency* can be more explicitly incorporated into SFG. This issue is discussed as part of the evaluation of SFG in Section 10.2.2.

3.3.2 Situated texts and multilingual textuality

As well as developing a systemic functional grammar of German and an Upper Model for German derived from the Penman Upper Model for English (see Section 2.2.3), the KOMET group has developed an architecture for *situated* text generation. This is based on the SFG approach to text as situated in the three dimensions of *field*, *tenor* and *mode* as mentioned in Section 1.3. This SFG approach is combined with Rhetorical Structure Theory for text structure, and with an Explanatory Combinatorial Dictionary (derived from Mel'čuk's Meaning-Text Theory) for lexicalization decisions. This work, described in [Bateman *et al.* 1991a], is outside the scope of this thesis.

The Darmstadt group have also investigated fundamental issues in SFG-based multilingual generation. An SFG of Dutch [Degand 1993] has been added to the Nigel grammar of English and the KOMET grammar of German, and a unified Upper Model has been created from those of Penman and KOMET. The group is now extending its architecture for situated text generation to handle *multilingual textuality*. The issues of divergent textuality across languages are described in [Bateman *et al.* 1993], but are also outside the scope of this thesis.

Chapter 4

Machine translation

4.1 Mainstream MT

4.1.1 Commercial development in Japan

In Japan, several companies have worked on the development of machine translation systems over the past 10 years or so. Sharp Corporation, for example, have produced and are selling the DUET Qt system for translation in both directions between Japanese and English. This is a highly-developed practical commercial system using mainstream MT techniques. Documents can be input easily in either language via a very accurate OCR scanner. A general dictionary of around 80,000 words is augmented by large specialist dictionaries for information processing, electronics, economics and various other fields, and by the facility to develop the user's own dictionary. The MT system is implemented in C and runs on a portable (notebook-type) Unix system with a mouse-driven window interface. The translation throughput is around 12,000 words per hour. All these practical facilities are being steadily improved.

In the DUET Japanese-English system on which I worked, translation is based on a combination of syntactic dependency structures and semantic case frames. The basic unit of translation is a single sentence. Noun phrases and fragmentary input can be translated, but larger texts of more than one sentence are split into separate sentences which are translated independently of each other, so there is no processing of coreferentiality across texts. As Japanese does not generally specify definiteness or number in noun phrases, and generally drops subjects and objects, leaving them to be inferred from the context

by the reader, while English requires all of these to be made explicit, the English generation component has to make decisions for these features, based only on whatever information is available in the source sentence, without the context in which it occurs. Inevitably, this information is frequently insufficient to make the necessary decisions correctly, so post-editing is essential in cases where high-quality translation is required, as discussed in Section 4.1.2.

There are two approaches to solving the insufficiency of information for high-quality English generation. One is to extend the analysis of the source text from single sentences to larger units, ultimately the whole text. This approach is being followed in Sharp Corporation's Twintran research project [Jelinek *et al.* 1990]. The other approach is to obtain correct choices from a human user at some point, before, during or after translation. This approach is the one investigated in this thesis.

4.1.2 Differences between import and export translation

In the practical development of effective machine translation systems the direction of translation is fundamental. *Import* translation from foreign languages into the user's language should be contrasted with *export* translation from the user's language into foreign languages. This difference in direction determines what kind of MT system design is most effective.

In import translation the source text is normally in only one foreign language, but in export translation the same text often needs to be translated into several target languages. For example, a company which sells products throughout the European Community needs to translate the same product documentation into all the EC languages.

If the original documentation is in Japanese, and is to be translated into the EC languages, there is a major linguistic difference in the direction of translation. Translation into European languages from Japanese requires decisions about singular/plural and definite/indefinite for every noun phrase in every sentence, and decisions about pronouns to fill missing subjects and objects in nearly every sentence. These decisions require both text-wide linguistic analysis of the source text and factual knowledge about the real-world domain. However, most of the information required for these decisions is the same for all the EC languages.

Another difference is that import translation always starts with a source text, but export translation sometimes has no source text. For example, there are 2 approaches for a Japanese company needing

product documentation for the EC. One approach is to write the documentation in Japanese first, then translate from the Japanese into the 9 EC languages. The other approach is to write the documentation in both Japanese and European languages at the same time. These alternatives are discussed further in Section 4.3.

Another difference is that in export translation it is usually not so difficult to get further information about the subject matter to be translated. The company which wants the export translation is usually the same company which makes the products. So the people who do the export translation can ask the people who designed the products for extra information and real-world knowledge.

Above all, there is a crucial difference in the requirement for post-editing. The quality of the raw output of MT has reached a level at which, in import translation, the post-editing stage is no longer always necessary for comprehension. The user, without needing a knowledge of the source language, can read the rough translation in his own language, and may find it sufficiently useful as it is.

For export translation, however, the target text is typically intended for customers, and the raw output of current systems falls short of the required standard. So post-editing by a target language expert is essential. This extreme dependence on the human post-editor means that current systems are restricted, for export translation, to use by professional translation agencies.

4.2 Research directions in MT

4.2.1 MT for monolinguals: the Alvey project

The question of how to reduce the dependence on expert post-editing was made the central research theme of the UK Alvey programme's MT project from 1984 to 1987. The research prototypes were designed to be suitable for use by monolingual English speakers for both import and export translation of Japanese without post-editing. The project, described in [Wood & Chandler 1988], was therefore called "Read and write Japanese without knowing it".

Aidtrans: Japanese-English for English users

The Japanese-to-English part of the Alvey MT project was led by Jiri Jelinek at the University of Sheffield, with myself as the participant from ICL, the industrial partner. In this approach to import

translation for monolinguals, source text ambiguities were deliberately "explicated". Therefore very little disambiguation by semantic restrictions was included in the system. Multiple target text alternatives were generated, expressing all the ambiguities, and these were presented to the user for interactive selection based on the user's own domain knowledge. In this way, the user could act as post-editor, without knowledge of the source language.

The Sheffield Aidtrans project was restricted to single sentence translation, without text-wide contextual analysis. This led to massive overgeneration of alternatives. In the subsequent Protran and Twintran projects at Kobe University, with Sharp Corporation as industrial partner ([Jelinek *et al.* 1990]), the system was redeveloped with semantic preferences and text-wide coreferentiality analysis, to generate only the most plausible and textually coherent alternatives.

Ntran: English-Japanese for English users

The English-to-Japanese part of the Alvey MT project was led by Pete Whitelock at UMIST, with Brian Chandler from ICL. Recent grammatical theories from mainstream Computational Linguistics were used: Lexical Functional Grammar for English analysis, and Unification Categorial Grammar for Japanese generation.

In export translation for source language monolinguals, it is essential to decide between the possible alternative translations, and generate only a single target language sentence. Therefore techniques for interactive source disambiguation were developed which did not require a knowledge of the target language.

Structural ambiguities in the source text were again "explicated" by the analysis stage, not automatically disambiguated. The ambiguities were presented to the user with questions and paraphrases to assist interactive selection of the intended alternative, again based on the user's domain knowledge. This form of pre-transfer editing therefore removed the need for one of the normal tasks of the target language post-editor, the correction of the system's mistaken structural disambiguations.

Alternative lexical choices in the target language were handled by presenting source language paraphrases of the choices to the user for interactive selection. This again removed the need for one of the target language post-editor's tasks, the correction of the system's mistaken lexical choices. [Wood & Chandler 1988] contains a fuller description of Ntran.

4.2.2 MT without a source text

Following the Ntran project described in Section 4.2.1, a more radical approach has been developed at UMIST to solve the problem of dependence on target language post-editing, in order to achieve successful export MT for monolinguals. In Ntran, analysis of the source text was followed by an interactive disambiguation stage to check the source language-speaking user's interpretation of the source text prior to transfer. In the later approach, described in [Somers *et al.* 1990], interaction with the user is made the starting point of the whole system, and there is no source text analysis. This approach is therefore called "MT without a source text".

In this approach, as in SFG-based generation, the system initiates requests for functional choices based on the user's communicative intent. The functional choices are organized into a network of menus of increasing "delicacy" just as in a systemic grammar network. However, the realizations of the functional choices are not lexicogrammatical clauses and structures, but complete "canned text" documents. The documents, such as standard business letters, are pre-translated by human translators and stored by the system under a functional classification.

Traversal of the network of menus leads to a pre-stored text matching the functional requirements of the user. The source language version of this text is then shown to the user as a potential "source text". Certain restricted classes of items, such as numbers, dates, names and addresses, can be changed interactively by the user. If the modified source text is then accepted by the user as expressing the desired communicative intent, the pre-translated and pre-stored target language version is retrieved and this is the target text.

The texts are stored in several languages, so this approach enables a monolingual user to achieve successful multilingual export translation without post-editing. However, communication is necessarily restricted to a fixed set of frequently-required and standardised messages. If the system does not have a text which matches the user's communicative intent, there is no way to proceed.

4.3 Multilingual MT versus multilingual generation

As mentioned in Section 3.1, international organizations need to produce large quantities of equivalent administrative documents on a regular basis in a fixed set of official languages, and international com-

panies need to produce equivalent product documentation in various languages. These requirements are currently addressed by large-scale translation operations. The large quantities of documents to be translated, and the extremely high cost of the translation operations, justify the great interest in the possibilities for improving these operations by the use of machine translation and translator's aids, as shown by the EC's large-scale support for the Eurotra project. Yet it is not at all clear that multilingual MT, or in fact any *translation*-based approach, is actually the right solution for these requirements.

The source of information from which equivalent texts need to be produced in different languages is not always a finished primary "source text" in one of the languages. In the case of an international company's product documentation, as mentioned in Section 3.1, the information may come from a CAD database. Although such a database would normally itself be monolingual, and is a primary and authentic source of information, it is not a source text ready for translation. However, in the current translation-based approach, the product's operating instructions would first be written in one language, then this new source text would be translated into the required target languages by a separate translation operation.

Typically the language used for this new source text would be the same language as the design database, and this text of the operating instructions would be written with the full participation of the product designers. It would therefore be an accurate and authentic source text. However, the subsequent translation operation would typically take place separately, with little or no participation from the product designers. The resulting target language texts therefore depend on the translators' inevitably imperfect understanding of the product's design, which is based entirely on their interpretation of the information present in the text of the operating instructions in the first language.

Languages differ widely in what aspects of information are normally explicit in the surface forms of words and what aspects are normally implicit. So the source text of the operating instructions, although accurate and authentic within the norms of the first language, will typically not explicitly express all the information which needs to be explicitly expressed within the norms of the target languages, and will also include explicit expression of some information which does not need to be expressed in the target languages.

In the translation-based approach, this problem can be addressed by giving the translators access to the design database and to the designers. In the multilingual generation-based approach, all the target

language versions of the operating instructions would be generated directly from the information in the design database, with the interactive participation of the designers. Both approaches would still require post-editing of the target texts by target language experts, but the generation-based approach avoids first "filtering" the information through the surface forms of a source text and then having to return to the original sources to obtain the missing information.

4.4 A systemic functional approach to export translation

Structural versus functional approaches

Transfer-based machine translation systems tend to be "source structure bound", if the target structure is based too closely on the source structure. Interlingua-based machine translation systems tend to be "target structure bound", if the target structure is based on pre-defined canonical ordering which fails to express the functional sentence perspective of the source text. Instead, a functional approach to machine translation seeks to treat structure as simply the syntagmatic realization of paradigmatic choices in source and target functional systems.

A functional, but non-interactive, machine translation system would tend to be "source system bound", because the only available information comes from an analysis of the source text. This can reveal the choices made in the functional systems of the source language, but cannot decide all the choices to be made in the functional systems of the target language.

The demonstration prototype described in Part II shows an approach to generating target sentences by directly navigating the functional systems of the target language. Because the generation of the target sentence is driven entirely by the target grammar, there is no interference from source language structure. The results are still "target system bound", because the user can only make choices which are available in the systems of the given grammar network.

Inquiry semantics for export translation

In the inquiry semantics approach, described in Section 2.2.2, the grammar network is traversed and features are selected from the systems of the language. At each choice point, the generation component raises an inquiry, and obtains a response which enables it to decide which feature to select. This approach

seems to be particularly suitable for target text generation in export translation. The inquiries come from inside the generation component, and are a method for consulting other components of the complete system. In an export translation system, the other components should include a source text analysis component and a human user interface component.

To use this approach in machine translation, the purpose of the analysis of the source text would be to enable *preselections* to be made in the target language grammar network traversal. This depends on being able to establish correspondences between the results of the analysis and the target functional systems, which would be easier if the analysis was itself functional. An approach to the future development of functional SFG-based analysis is briefly discussed in Section 10.2.3. However, even a formal or semantic analysis could be used to make some preselections. Other functional choices in the target language would still be made by user interaction, including those where the necessary information is not in the source text at all.

A new interpretation of "pivot language"

In multilingual translation systems, structural transfer rules are multiplied, so an interlingual or pivot approach becomes attractive. Some of the Japanese companies faced with the challenge of translation from Japanese into the languages of the European Community have already developed Japanese-English MT systems. To avoid the immense practical difficulties of building separate MT systems for each language-pair (Japanese-Portuguese, Japanese-Greek, ...), it makes sense for such a company to use English as a "pivot" language.

In the simplest form of the pivot approach, the Japanese documentation would be translated to English, using the existing Japanese-English MT system. Then the English version would be translated into the other 8 EC languages, using some European multilingual MT system, perhaps of the type pioneered by the Eurotra research project. Unfortunately, there are likely to be many problems with this simple approach if the MT systems are based on structural transfer. The problems already encountered in a single translation are likely to be compounded by the double translation.

A less simple approach might use an internal representation of the English text (similar to the Interface Structure in Eurotra, described in [Steiner *et al.* 1988]) as a pivot. Such an approach might indeed be sufficient for the requirements of import translation.

For Japanese-European export translation, however, the dependence on post-editors is an especially difficult problem, due to the severe shortage of people with knowledge of both Japanese and the various European languages. Therefore the approach of the demonstration prototype, based on a functional interface to a systemic grammar for generation, is particularly relevant as a way to reduce the post-editing load.

In a multilingual application of this approach, all the EC target languages would share a common systemic grammar. Most of the functional systems would be shared, but some would be unique to specific languages, as described in Section 3.2.2. The realization rules need not be shared. The Japanese user would make functional choices in the target systems, using a Japanese interface.

English can be used in the role of a pivot language in this approach. Most Japanese users know some English, but very few know other European languages. The English version of the generated target text, and not the other European language versions, can be displayed to the Japanese user for checking and interactive revision. The systemic choices made for English would be re-used in the generation of the other European languages, with relatively few further choices specific to the other languages.

Part II

The demonstration prototype

Chapter 5

The Mini-grammar

The grammar used for the demonstration prototype is taken from [Fawcett 1988]. This is a systemic mini-grammar for generation of English auxiliary verbs. It is deliberately restricted to coverage of a small area: English modal and auxiliary verbs. Therefore the only noun phrases included are trivial ones - proper names - with no attempt to include a normal system network for noun phrases, no adjectives, no determiners or quantifiers, no relative clauses. There are also no circumstantial adjuncts, no adverbs or prepositional phrases (except "by" phrases for agents in passives). On the other hand, for the focussed area of modal and auxiliary verbs, the mini-grammar has a fairly wide coverage of modality, tense, aspect, theme and mood, enabling generation of complex verb phrases such as in "Mark oughtn't to have been being kissed".

The mini-grammar was therefore selected as a convenient, small example of a working SFG grammar, and was adopted unchanged as a ready-made basis for the demonstration prototype. The implementation of the mini-grammar, however, described in Chapter 6, is completely new and is not based in any way on the implementation of Genesys in Cardiff. The requirements of the demonstration prototype - an X Window interface, Japanese menus, display of feature summary tables for modification of choices, backtracking and regeneration for sentence revision, and multilingual realizations - were quite different from those of the COMMUNAL project, so a new design and implementation was required.

5.1 The grammar network

The mini-grammar is a fragment of the full Genesys grammar described in Section 2.3.1, extracted and modified to be free-standing. Probability values are expressed in a simplified scale from 0 to 10. Some features which would lead into other areas of the full grammar are inhibited in the mini-grammar by being assigned zero probability, while other features are assigned absolute preference (i.e. 10).

There are two entry points for the mini-grammar network. The main entry point, which is the starting point for every generation with the network, has the entry condition *situation* and two output features, as shown in Figure 5.1. *Congruent* leads into the rest of the network for generation of a full clause, via 5 simultaneous systems, TRANSITIVITY, CONTINUING PERIOD, MOOD(1), POLARITY and INFORMATION FOCUS, which are described below. In a fuller grammar *non-congruent* would be used for generation of a nominalized situation, but in the mini-grammar the non-congruent choice is inhibited by setting its probability to zero.

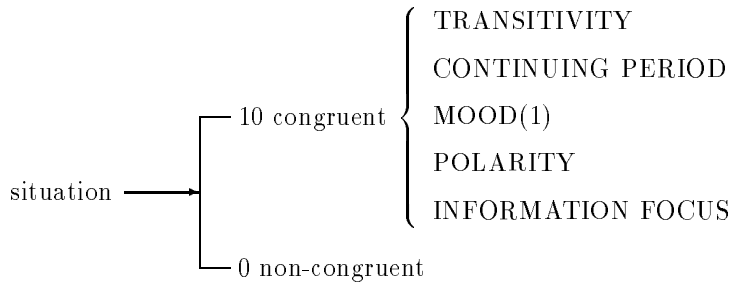

Figure 5.1: The initial entry to the mini-grammar

The other entry point for the mini-grammar, which in a fuller grammar would be the starting point for generation of noun phrases, has the entry condition *thing*. In the mini-grammar, all choices in the subnetwork for thing are inhibited except those which generate a proper name. The subnetwork for thing is only entered by a *re-entry* realization rule from the main network for situation, in order to generate a proper name noun phrase to fill a participant role in the clause.

The TRANSITIVITY region of the network is shown in Figure 5.2. Only material and relational process types are covered in the mini-grammar, so the choice of mental process type is inhibited by a zero probability value. Material processes are restricted to a choice of 5 verbs in the CULTURAL CLASSIFICATION system.

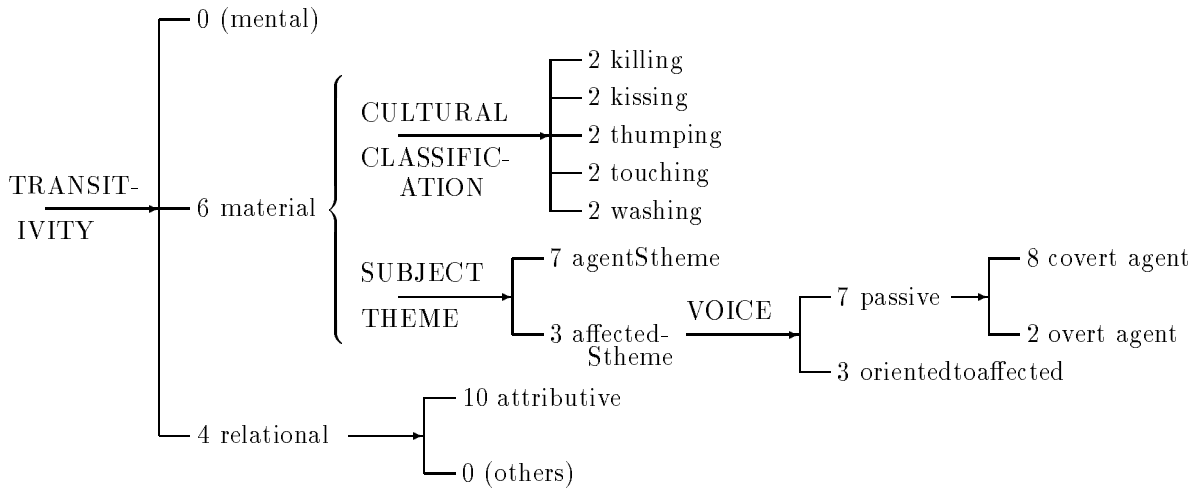

Figure 5.2: The TRANSITIVITY network

The MOOD network is shown in Figure 5.3. The output feature *information* is the entry condition for 3 simultaneous systems, REFERENCE POINT LOCATION, RELEVANT PASTNESS and MOOD(2). This part of the network handles tense and aspect. The Nigel grammar, following Halliday's mainstream analysis, divides primary tense into past, present and future, as described in detail in [Matthiessen & Bateman 1991] Chapter 7. However, Halliday also recognises an alternative analysis in which future tense is regarded as part of modality (i.e. a disposition to do something), and this is the view adopted in Genesys and the mini-grammar.

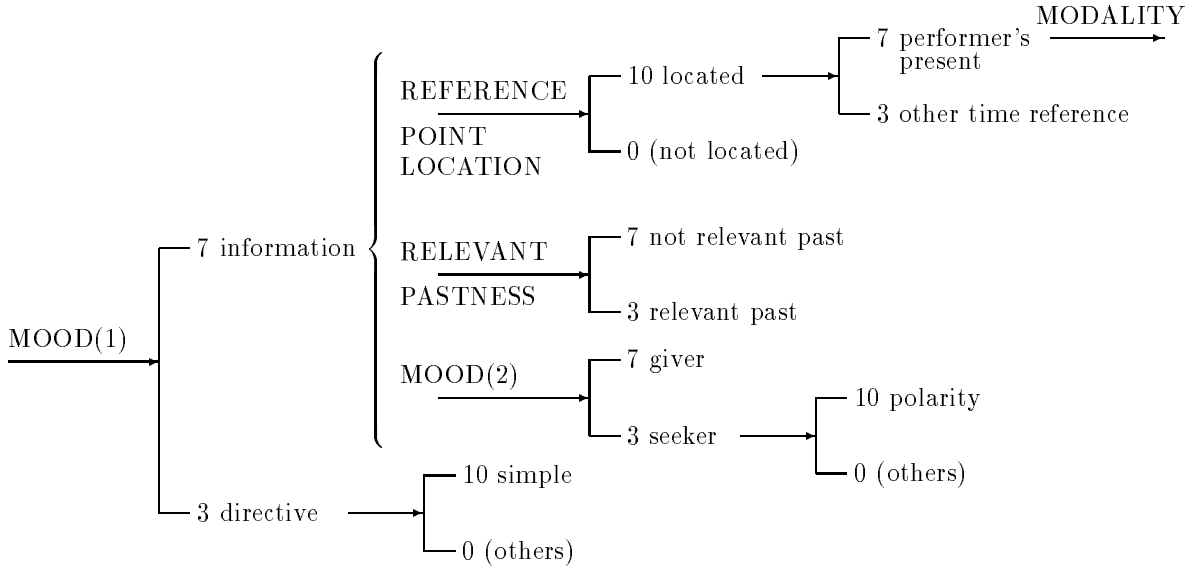

Figure 5.3: The MOOD network

REFERENCE POINT LOCATION therefore leads to only two choices in the *located* system: *performer's present* or *other time reference* (i.e. past). If the reference point is located at performer's

present, the MODALITY network, shown in Figure 5.4, is entered, and if *unconditional disposition* is selected, a future tense form is generated. RELEVANT PASTNESS, despite its name, handles aspect not tense: the choice of *relevant past* generates a perfective aspect form. The realization of these tense, aspect and modality combinations is described in Section 5.2.

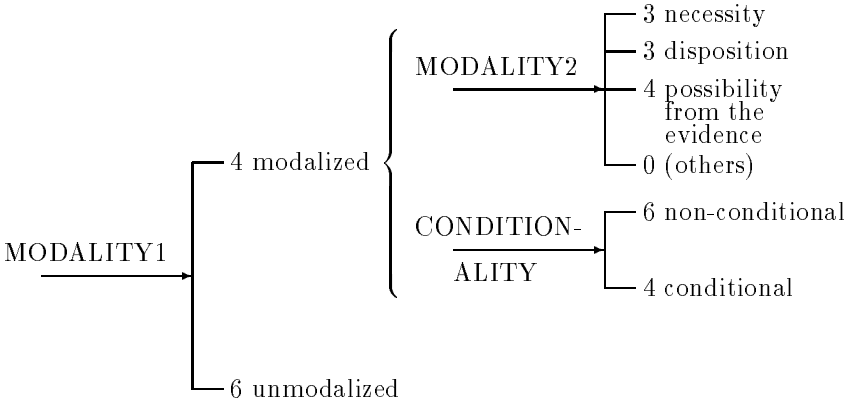

Figure 5.4: The MODALITY network

The CONTINUING PERIOD network, together with RELEVANT PASTNESS in the MOOD network, also handles aspect. CONTINUING PERIOD distinguishes progressive and non-progressive aspect. It consists of the single system shown in Figure 5.5.

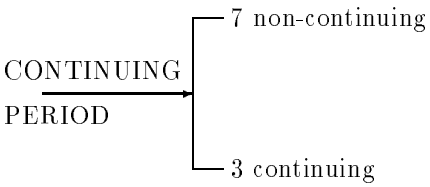

Figure 5.5: The CONTINUING PERIOD network

The POLARITY network consists of a single system, shown in Figure 5.6.

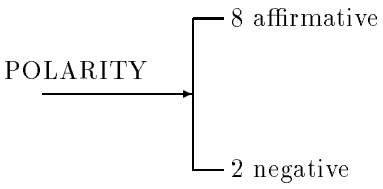

Figure 5.6: The POLARITY network

The INFORMATION FOCUS network, shown in Figure 5.7, has a system to select between marked and unmarked information focus, i.e. whether special emphasis should be placed on a particular item.

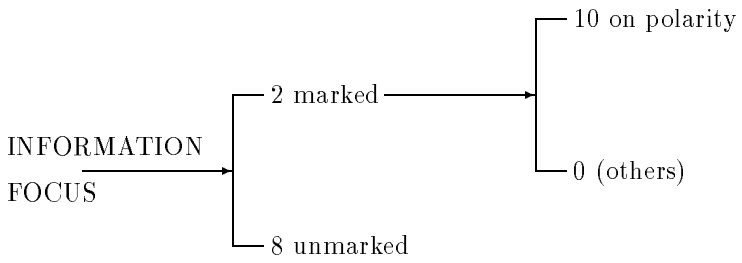

Figure 5.7: The INFORMATION FOCUS network

In the demonstration implementation, the emphasis is shown by displaying the marked item in upper case. In the mini-grammar, information focus can only be marked on polarity, other candidates for marked focus (such as Agent or Process) being inhibited by a zero probability value.

5.2 The realization rules

The realization rules in the mini-grammar are listed in full in [Fawcett 1988]. They take the form already described in Section 2.3.1 for the full Genesys grammar. The use of complex conditions on the left-hand side of the rules can be seen from the following examples.

Forms of the auxiliary "have" for perfective aspect are realized according to the following complex conditions: if [relevant past & NOT modalized & performer's present] then "has", if [relevant past & NOT modalized & other time reference] then "had", if [relevant past & modalized] then "have".

Forms of the auxiliary "will" for dispositional modality (including future tense) are realized according to the following complex conditions: if [disposition & conditional] then "would", if [disposition & non-conditional & affirmative] then "will", if [disposition & non-conditional & negative] then "wo" (in the mini-grammar the negative feature is realized as the suffix "n't"). These realizations are discussed further in Section 10.1.

As mentioned in Section 2.3.1, the *starting structure* for Clause in the mini-grammar has 9 positions. The *insertion* realization operation puts an element of structure into one of these positions. The element realizing Subject is inserted at position 2 by the realization rule "S @ 2". The insertion position of the finite auxiliary, referred to as Operator, depends on the output feature of the MOOD(2) system: if the complex condition includes *seeker*, Operator is inserted at position 1 by the operation "O @ 1", but if the complex condition includes *giver*, Operator is inserted at position 3 by "O @ 3". Since Subject is inserted at position 2, this mechanism achieves the required ordering of Subject and Operator.

Chapter 6

The Prolog implementation

6.1 Representation of the grammar network

Systemic grammars have traditionally been presented in the form of systemic network diagrams, using graphical conventions which are not suitable for direct input to linear computer files. For computational implementation, the Nigel grammar was written in a Lisp-like formalism, shown in Section 2.2, and the Genesys grammar was written in a Prolog-like formalism, shown in Section 2.3.1.

The mini-grammar used for this prototype demonstration is extremely small, so a naive approach to a grammar-writing formalism is sufficient. A notation for writing systemic grammar networks was implemented simply by declaring a set of Prolog operators. Operator symbols were chosen which are similar to the graphical conventions normally used in writing systemic grammar networks.

Choice systems

The most important items in a systemic grammar are the disjunctive choice systems, represented in the traditional network notation by a right-angled branching tree turned on its side, as in Figure 6.1.

In Figure 6.1 'clause' is the entry condition, 'MOOD TYPE' is the name of the system, 'imperative' and 'indicative' are the alternative output features, exactly one of which must be chosen.

The Prolog operator declared to represent the disjunctive choice system is the left square bracket '['. This is a binary infix operator. The entry condition is written on the left of the operator. The alternative choices are written on the right of the operator in the form of a Prolog list. The system in

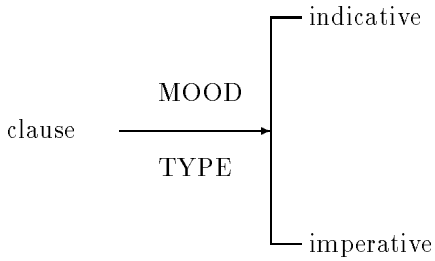

Figure 6.1: Disjunctive Choice System Representation

Figure 6.1 is therefore written as:

```
clause '[' [indicative, imperative].
```

Dependencies between systems

In the conventional systemic grammar network notation, dependency between systems is shown by their spatial arrangement in the graphical representation. When an output feature of one system is the entry condition for another system, the second system's entry point is simply aligned with the first system's output feature. So in Figure 6.2 'indicative' is both an output feature of the MOOD TYPE system and the entry condition of the INDICATIVE TYPE system.

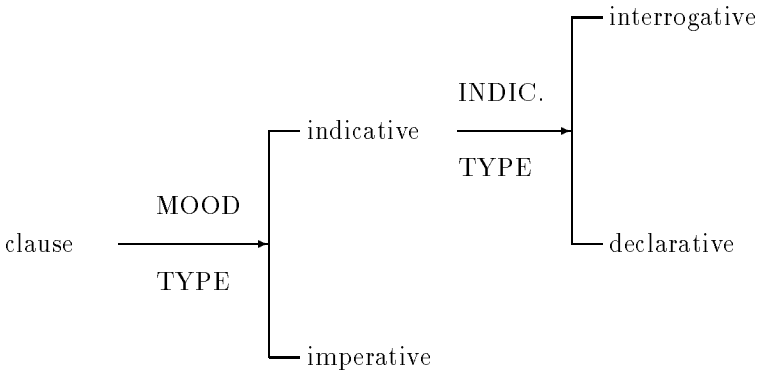

Figure 6.2: System Dependency Representation

In the simple Prolog notation, the dependency is represented by using the same feature name in both systems, as a member of the output feature list of the first system, and as the entry condition of the second system. So the dependency shown in Figure 6.2 is written as:

```
clause '[' [indicative, imperative].

indicative '[' [declarative, interrogative].
```

Simultaneous systems

Selection of an output feature in a choice system may cause simultaneous entry to two or more parallel systems. This is represented in the traditional network notation by a left curly bracket, which vertically spans the entry points of the simultaneous systems on its right, as in Figure 6.3.

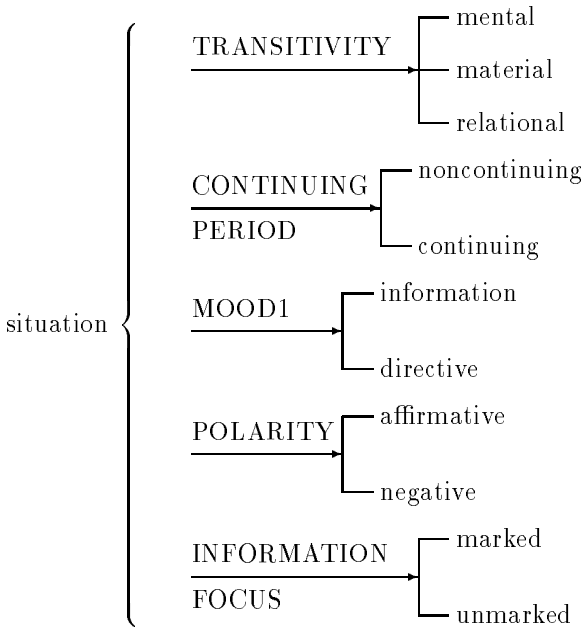

Figure 6.3: Simultaneous Systems Representation

The Prolog operator declared to represent parallel simultaneous systems is the left curly bracket '{'. This is a binary infix operator. The entry condition is written on the left of the operator. The simultaneous systems are written on the right of the operator in the form of a Prolog list. The parallel systems in Figure 6.3 are therefore written as:

```
situation '{' ['TRANSITIVITY', 'CONTINUINGPERIOD', 'MOOD1',
               'POLARITY', 'INFORMATIONFOCUS'].

'TRANSITIVITY' '[' [mental, material, relational].

'CONTINUINGPERIOD' '[' [noncontinuing, continuing].

'MOOD1' '[' [information, directive].

'POLARITY' '[' [affirmative, negative].

'INFORMATIONFOCUS' '[' [marked, unmarked].
```

Compound and alternative entry conditions

Compound and alternative entry conditions occur comparatively rarely in systemic grammar networks. They do not occur at all in the demonstration mini-grammar, but are included in the Prolog notation for completeness.

In the traditional diagram notation, compound entry conditions are represented by right curly brackets and alternative entry conditions are represented by right square brackets. The dependencies are represented by drawing lines between the dependent systems. Typically these dependency lines cross the diagram diagonally and look untidy.

The Prolog operator declared to represent compound entry conditions is the right curly bracket '}'. This is a binary infix operator. The compound entry conditions are written on the left of the operator in the form of a Prolog list. The system which is entered is written on the right of the operator.

The Prolog operator declared to represent alternative entry conditions is the right square bracket ']'. This is a binary infix operator. The alternative entry conditions are written on the left of the operator in the form of a Prolog list. The system which is entered is written on the right of the operator.

6.2 Representation of the realization rules

In the systemic grammar literature, while established conventions have emerged for the representation of system networks, realization rules have been represented in a wide variety of notations. Realization has most often been indicated by a diagonal arrow $\searrow$ symbol. The Nigel grammar uses a Lisp-based notation. The Communal grammar uses a set of realization operator symbols.

Some systemic grammars include the realization rules as part of the system network. When an output feature has a realization associated with it, the realization is written next to the output feature in the system network. The Nigel grammar uses this approach.

Other systemic grammars gather the realization rules into a separate place after the system network, and provide some means of relating the realization rules to the features of the network. The Communal grammar uses this approach. The Communal grammar also includes conditions inside the realization rules. A realization rule only applies if the conditions are true.

In the mini-grammar demonstration, a simple notation for writing realization rules has been im-

plemented by declaring a set of Prolog operators, based loosely on the notation used in the published version of the mini-grammar [Fawcett 1988].

The format of the realization rules is in three parts: the name of the feature with which the realization is associated, the set of conditions for application of the rule, and the set of realization operations to be performed. This format is therefore similar to the form of production rules in an expert system. The feature name forms the name of the rule. The set of conditions forms the left-hand side of the rule. The set of operations forms the right-hand side of the rule.

The general format of a realization rule is therefore:

```
realization(featurename, [list of conditions], [list of actions]).
```

An example of an actual realization rule is:

```
realization(agentStheme,  [directive],  [complement '@' 9,
                                         affected '/' complement,
                                         affected 'RE' thing]).
```

Insertion

The insertion operation, which inserts an element of structure at a specific position in a starting structure, is represented by the Prolog operator '@'. This is a binary infix operator. The element is written on the left of the operator. The position is written on the right of the operator.

For example, to insert the element 'complement' at position 9:

```
[complement '@' 9]
```

Conflation

The conflation operation, which conflates functions from two different metafunctions into a single element of structure, is represented by the Prolog operator '/'. This is a binary infix operator. The two functions are written one on each side of the operator.

For example, to conflate the functions 'affected' and 'complement':

```
[affected '/' complement]
```

Exponence

The exponence operation, which expounds an element of structure by an item (such as a word), is represented by the Prolog operator '<'. This is a binary infix operator. The element being expounded

is written on the left of the operator. The item which expounds it is written on the right.

For example, to expound the element 'relevantpastauxiliary' by the item 'has':

```
[relevantpastauxiliary '<' has]
```

Re-entry

The re-entry operation, which causes an element of structure to be realized by re-entry to the system network at a specified point, is represented by the Prolog operator 'RE'. This is a binary infix operator. The element to be realized is written on the left of the operator. The entry point at which re-entry is to be made is written on the right.

For example, to realize the element 'affected' by re-entering the network at entry point 'thing':

```
[affected 'RE' thing]
```

Condition Negation

The negation of a condition, in the condition part of a realization rule, is represented by the Prolog operator 'NOT'. This is a unary prefix operator.

For example, to negate the condition 'directive':

```
['NOT' directive]
```

6.3 Grammar network traversal procedures

The system network is traversed by a simple recursive algorithm. The fundamental parts of the algorithm are the 'traverse' procedure, which handles navigation across the network, and the 'choose' procedure, which handles choices in disjunctive choice systems.

The 'traverse' procedure

The 'traverse' procedure controls traversal of the system network, based on the representation of dependencies between systems and of simultaneity of systems.

The basic part of the procedure deals with one disjunctive choice system. Given an entry condition, if that entry condition is included in the network representation with the disjunctive choice operator and a list of alternative output features, the 'traverse' procedure calls the 'choose' procedure to select one

of the output features, appends the chosen feature to the set of features selected so far, and recursively calls itself with the newly chosen feature as a new entry condition.

This fundamental part of the algorithm is written in Prolog as:

```
traverse(Mode, Entry, SetIn, SetOut) :-
        Entry '[' List,
        not member(Entry:_, SetIn),
        choose(Mode, Entry, List, Chosen),
        append(SetIn, [Entry:Chosen], SetChosen),
        traverse(Mode, Chosen, SetChosen, SetOut).
```

Simultaneous systems should be traversed logically in parallel, but in ordinary Prolog are in fact traversed procedurally in sequence. The set of simultaneous systems is represented in the network as a Prolog list, and the 'traverse' procedure calls a tail-recursion procedure 'traverseAll' to process the list in sequence:

```
traverse(Mode, Entry, SetIn, SetOut) :-
        Entry '{' List,
        traverseAll(Mode, List, SetIn, SetOut).
traverseAll(Mode, [], SetIn, SetIn).
traverseAll(Mode, [H|T], SetIn, SetOut) :-
        traverse(Mode, H, SetIn, SetH),
        traverseAll(Mode, T, SetH, SetOut).
```

The 'choose' procedure

The 'choose' procedure works differently in menu-driven mode and in random mode. In menu-driven mode it calls a procedure to present the list of alternative output features as a screen menu using X Windows. The feature selected interactively by the user is returned and passed back to the 'traverse' procedure.

In random mode, the 'choose' procedure itself selects an output feature from the list, using the probability values included in the list, together with a random number generated by the Prolog built-in function 'random'. This probabilistically selected feature is passed back to the 'traverse' procedure.

The selection expression

As the 'traverse' procedure traverses the system network, the selected output features are appended to a list. This list constitutes the *selection expression* from which the sentence structure is generated by the realization component (the structure realizer and the realization rules).

The selection expression consists of a list of system:choice pairs, similar to the sets of attribute:value pairs used to represent feature structures in unification-based grammars. Its format is simply:

```
['SYSTEM1':choice1, 'SYSTEM2':choice2, ...]
```

6.4 Realization procedures

The realization procedures generate the output sentence structure from the selection expression by two logically distinct processes, which can be called *functional realization* and *structural realization*. The distinction is based on that explained in [Berry 1977] (see Section 1.2.2).

Functional realization is performed by the 'realize' procedure, and structural realization is performed by the 'realizeStructure' procedure. The whole realization process is controlled by the 'realizeAll' procedure, which uses tail recursion to call 'realize' for each output feature in the selection expression, then calls 'realizeStructure' once. 'realizeAll' is written in Prolog as:

```
realizeAll(Features, Functions, Structure) :-
        [First|Rest] = Features,
        realizeRest([First|Rest], Features, Functions, Structure),
        realizeStructure(Functions, Structure).

realizeRest([_:Value|Rest], Features, Functions, Structure) :-
        realize(Value, Features, Functions, Structure),
        realizeRest(Rest, Features, Functions, Structure).
realizeRest([],_,_,_).
```

The procedure 'realize' is called for each output feature value. It uses the realization rule for the feature to obtain the list of conditions and the list of actions, then checks the conditions, and if they are all true, does the actions. It is written as:

```
realize(Value, Features, Functions, Structure) :-
        realization(Value, Conditions, Actions),
        checkConditions(Conditions, Features),
        doActions(Actions, Features, Functions, Structure).

checkConditions([], Features).
checkConditions([First|Rest], Features) :-
        checkCondition(First, Features),
        checkConditions(Rest, Features).

checkCondition('NOT' Value, Features) :-
        not member(_:Value, Features).
checkCondition(Value, Features) :-
        member(_:Value, Features).
```

```
doActions([], Features, Functions, Structure).
doActions([First|Rest], Features, Functions, Structure) :-
        doAction(First, Features, Functions, Structure),
        doActions(Rest, Features, Functions, Structure).
```

Functional realization

Functional realization creates a set of *functions* using the selection expression and the realization rules. The set of functions is unordered and sideways-open, and is processed by DAG unification [Gazdar & Mellish 1989] using the 'unify' procedure. The following 'doAction' clauses handle the realization operations of insertion, exponence, conflation, and re-entry:

```
% Insertion
doAction(Element '@' PlaceNum, Features, Functions, Structure) :-
        unify(Functions, [Element:[place:PlaceNum|_]|_]).

% Exponence
doAction(Element '<' Word, Features, Functions, Structure) :-
        unify(Functions, [Element:[base:Word|_]|_]).

doAction(Element '<+' Suffix, Features, Functions, Structure) :-
        unify(Functions, [Element:[base:Word, suffix:Suffix|_]|_]).

% Conflation
doAction(Role '/' Element, Features, Functions, Structure) :-
        unify(Functions, [Element:X|_]),
        unify(Functions, [Role:X|_]).

% Re-entry
doAction(Role 'RE' Entry, Features, Functions, Structure) :-
        unify(Functions, [Role:RoleList|_]),
        enter(unknown_mode, Entry, [], RoleFeatures, RoleStructure, Role),
        unify(RoleList, RoleStructure).
```

Structural realization

Structural realization uses a pre-defined *starting structure* and fills the places in the starting structure with functions from the set created by functional realization. In the demonstration mini-grammar, the basic predefined starting structure is for *clause* and consists of 9 numbered places. It is created by:

```
doAction(clause, Features, Functions, Structure) :-
        unify(Structure, [clause:[1:Place1,2:Place2,3:Place3,4:Place4,
                  5:Place5,6:Place6,7:Place7,8:Place8,9:Place9|_]|_]).
```

The places in the starting structure are filled by the following procedure:

```
realizeStructure(Functions, Structure) :-
        Structure = [clause:PlaceList|_],
        fillPlaces(Functions, PlaceList).
realizeStructure(Functions, Structure).

fillPlaces(Functions, []).
fillPlaces(Functions, [N:StructurePlaceN|RestStructurePlaces]) :-
        member(Element:ElementList, Functions),
        unify([place:N|_], ElementList),
        delete(place:N, ElementList, RestElement),
        unify(StructurePlaceN, Element:RestElement),
        fillPlaces(Functions, RestStructurePlaces).
fillPlaces(Functions, [N:PlaceN|RestPlaces]) :-
        fillPlaces(Functions, RestPlaces).
```

Sentence formatting

The sentence structure is finally converted into a surface string by a finishing procedure which processes suffixes and handles capitalization, spacing and punctuation.

Chapter 7

The Japanese interface

7.1 Design of the menus and feature summaries

The purpose of the Japanese interface is to enable a Japanese-speaking user to make choices in the system network in order to generate English sentences. Typically, the Japanese user would have some knowledge of English, but would have great difficulty in using complexes of auxiliary and modal verbs correctly.

Because the generation is controlled by the systemic mini-grammar, the inquiry semantics approach guarantees that a coherent and complete set of choices will always be made, and the realization rules guarantee that the generated sentences using these choices will always be well-formed. The problem is how to present the functional choices for English to the Japanese user.

The menu buttons

It would not be very helpful to present the choices using terminology based on surface syntactic distinctions in English, as this is precisely the area of difficulty for the intended user. However, one reason for selecting the mini-grammar from the Cardiff COMMUNAL project is that the labelling of the choices within the grammar is already semantically and functionally orientated rather than syntactically orientated, as discussed in Section 2.3.1. Therefore the choice labels themselves are used in the interface.

Of course, the labels in the mini-grammar are in English, so equivalent labels in Japanese had to be created. Since the intended user has some knowledge of English, both the Japanese and the English

labels are used together in the interface, in the form of bilingual X Window buttons. An example menu is shown in Figure 7.1. The buttons are labelled "giver" and "seeker".

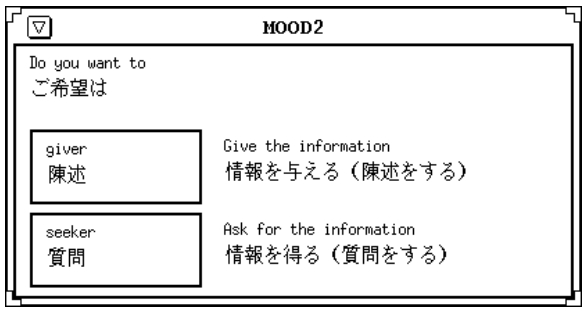

Figure 7.1: An example menu

The menu questions

The labels alone would not make the functional choices sufficiently clear. Therefore short questions asking about the user's communicative intent were added. It was intended that these questions would be based directly on the felicity conditions used in the COMMUNAL system, as described in Section 2.3.2. However, felicity conditions are not given for the published mini-grammar, so the questions had to be created for the demonstration prototype. The questions were created in both Japanese and English, and both versions are used together in the interface. In the example menu shown in Figure 7.1, the question asks whether the user wishes to give or seek the information.

The feature summaries

When the grammar network traversal is finished, the set of selected feature values is used by the realization rules to generate the structures and words of the English sentence. The generated sentence is displayed (in English only) with a summary table of the features which it realizes. The summary table shows the feature names in both English and Japanese. An example of the display of the generated sentence and the feature summary table is shown in Figure 7.2.

In the summary table, the left-hand column shows the names of the features and the right-hand column shows their values. The length of the table is variable, as the number of features collected into the selection expression on any specific traversal of the network depends on the route taken: some choices lead to many further (more delicate) choices in complex regions of the network, while other

Figure 7.2: An example sentence

choices do not lead into those regions.

The X Window System

The Japanese interface was implemented in the X Window System. There are two main reasons for using this system. First, the user can make his functional choices very easily, simply by clicking the mouse pointer on the desired menu item. Second, the summary of the selected feature values can itself be used as a menu, to allow easy revision of the choices. This is possible because in the X Window System each item in the table of features is a separate window, so it can be selected for revision by

simply clicking the mouse pointer on it.

Other reasons for using X Windows are that it can support mixed Japanese and English displays (and accented European characters for multilingual generation), and that further help information can be added and displayed easily by means of pop-up help texts.

The C implementation

In the demonstration prototype, the systemic grammar network and realization rules are written in a Prolog-based notation, and the grammar network traversal and structure realization programs are implemented in Prolog, as described in Chapter 6. However, the Japanese interface using the X Window System is implemented in the C programming language.

Some versions of Prolog now provide direct interfaces to X Windows. However, the implementation was done in YAP Prolog without such a direct interface. A set of bindings was defined between the YAP Prolog system and the C functions which manipulate the X Window displays. Because of the need to switch between Japanese and European fonts, the lower-level Xlib interface to the X Window System was used, rather than the higher-level Xt toolkit interface.

7.2 Using the interface

To generate a new sentence

When starting to generate a new sentence, the grammar network is always entered at "situation" (see Section 5.1 for an explanation of the mini-grammar). This leads to menus for 5 parallel systems (TRANSITIVITY, CONTINUINGPERIOD, MOOD, POLARITY, INFORMATIONFOCUS). Choices from all these menus will be required, but subsequent menus are only presented when previous choices lead to them. So the requests for functional choices are driven entirely by the systemic grammar of the target language (English).

Alternatively, the choices at each system choice point in the grammar network may be made probabilistically. In this mode, probability values written in the grammar network are used with random numbers to traverse the network. The randomly generated sentence is displayed with its feature summary table, as in menu-driven generation, and can be revised in the same way.

A future possibility is a combination of probabilistic and menu-controlled feature selection modes, in which the machine will start to generate a stream of English sentences in the probabilistic mode. As the user gradually makes functional choices by menu, these will replace the probabilistic ones, and the stream of sentences will gradually become less random, and converge towards the expression of the user's communicative intent.

To revise an existing sentence

The feature summary table may be used for iterative revision. It is an array of subwindows, so it can be used directly as a menu. Any feature to be revised may be selected simply by clicking on its subwindow with the mouse. The bilingual functional choice menu for this feature is then presented again, and a new value for the feature may be chosen from it.

This choice may lead to further choices from subsystem menus, until the network traversal is completed. Then the realization rules generate a new sentence, which realizes the same features as the old sentence, except for the changed feature and any subfeatures.

7.3 Illustration of monolingual sentence revision

Given the generated sentence displayed in Figure 7.2, the user might wish to remove the explicit reference to the agent of the action. By clicking the mouse pointer on the AGENTOVERTNESS box in the summary table, the AGENTOVERTNESS choice system will be re-entered and the Agent Overtness menu shown in Figure 7.3 will be presented.

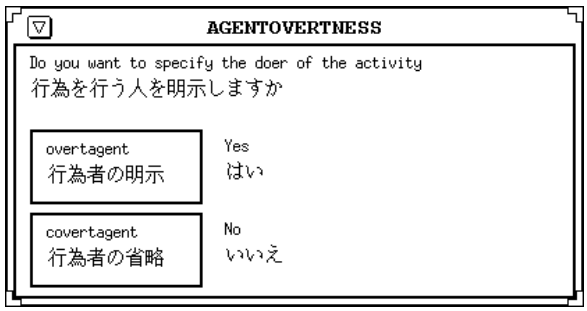

Figure 7.3: The Agent Overtness menu

If the user selects covertagent instead of overtagent, no further choices are required in the net-

work traversal, so the realization procedures will immediately generate the revised sentence. It will be displayed with its summary table, as shown in Figure 7.4.

Figure 7.4: The sentence after revision

After the new sentence is displayed, another feature may be revised. If the user clicks on SUBJECTTHEME in the new summary table, the SUBJECTTHEME system will be entered and the Subject Theme menu shown in Figure 7.5 will be presented.

If the user now selects agentStheme instead of affectedStheme, another revised version of the sentence will be generated, as shown in Figure 7.6. This revision loop may be repeated as often as desired.

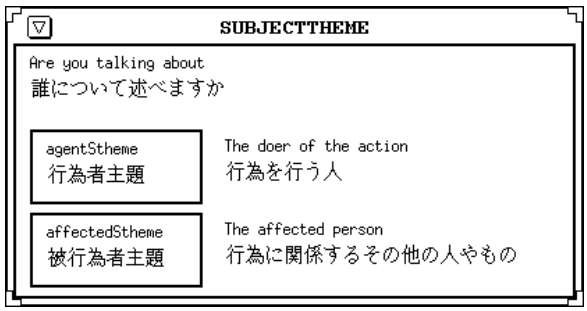

Figure 7.5: The Theme menu

Figure 7.6: The sentence after further revision

Chapter 8

The multilingual version

8.1 Adding French and German

After the small demonstration prototype for generation of English auxiliary verbs had been implemented as described in the previous chapters, it was extended to add generation of French and German in parallel with English.

The intention was to give a practical demonstration that most of the feature choices needed for generation of European languages are common across at least most of the main EC languages. The importance of this commonality is that the cost of making the choices (whether by user interaction or by any other means within the inquiry semantics approach) is rewarded by a greater pay-off if the same choices can be used to generate several languages rather than just one, as discussed in Section 4.4.

More specifically, it was intended to demonstrate practically that if the interactive Japanese interface can be used successfully by a Japanese user (with some limited knowledge of English) to make the choices necessary for English, then with relatively few further choices the Japanese interface can be used successfully by a Japanese user (with some limited knowledge of English but no knowledge of other European languages) to generate several European languages in parallel. This is based on the new interpretation of the old idea of using English as a pivot language for machine translation, as discussed in Section 4.4.

Shared system network

In fact, the ideas for shared multilingual system networks, discussed in Section 3.2.2, have not (so far) been implemented in the extension of the small demonstration prototype from monolingual English generation to multilingual English, French and German generation. Within the narrow range of linguistic phenomena covered by the mini-grammar, the English network can be used unchanged for French and German with very few problems.

Some systems in the shared network are not relevant for all three languages. For example, the English distinction between progressive and simple present is redundant for French and German (except when it is pragmatically important enough to be marked by adverbial adjuncts), as shown in Figures 8.3 and 8.4. However, the multilingual version of the demonstration prototype does not include any facility for selecting subsets of languages to be generated: all three languages are always generated in parallel. Therefore the progressive/simple decision must be made (for present tense sentences) - in fact any choice needed for any one language must be made if its entry condition is satisfied.

One linguistic problem which arises in the multilingual use of the English mini-grammar is in subject-verb gender agreement in French. All the noun phrases in the mini-grammar are deliberately restricted to third-person singular proper names, so that subject-verb agreement is trivially always third-person singular. But some of the names are feminine, and in French passives the passive participle must agree in gender with the subject. This issue is discussed further in Section 9.1.

Separate monolingual realization rules

Although the system network is shared, completely separate sets of monolingual realization rules were written for English, French and German. The English rules were unchanged from the monolingual version, described in Section 5.2. The French rules were based closely on the English rules, as the French word order and use of auxiliaries is fairly similar to the English. The German rules required considerable changes, as the German word order is significantly different.

Shared interactive Japanese interface

Because the system network is shared, and the Japanese interface is used solely for navigation through the system network, the same Japanese interface is also shared. The same selection expression, produced

by the network traversal, is used for the realization of the three languages by the three separate sets of realization rules.

Apart from the addition of two further windows in the top area to display the French and German sentences as well as the English, the only modification required in the X Window interface component is to handle the display of accented characters for French and German. In order to do this, the default ascii font used in the monolingual English version is replaced with an ISO-8859 "Latin" font. This font is used for English as well as French and German, so that the same C code displays all the three languages.

8.2 Illustration of multilingual sentence revision

Figure 8.1: Multilingual generation (1)

Figure 8.1 shows an example result of a generation by the multilingual demonstration prototype. The table of selected features is displayed in exactly the same way as in the monolingual English version, but the parallel English, French and German sentences are all displayed in the top area.

Figure 8.2: Multilingual generation (2)

Revision of the three sentences is also performed in exactly the same way as in the monolingual version, because the revision is done at the shared functional level, not at the surface realization level. If the user selects the SUBJECTTHEME window, and selects affectedStheme from the SUBJECTTHEME menu which is presented, and then selects passive from the VOICE menu which follows it automatically, followed by overtagent from the AGENTOVERTNESS menu which follows, all three sentences will be

regenerated and displayed as in Figure 8.2.

If the user now selects the MOOD2 window, and selects seeker from the MOOD2 menu which is presented, the sentences will be regenerated and displayed as in Figure 8.3.

Figure 8.3: Multilingual generation (3)

Finally, if the user selects the CONTINUINGPERIOD window, and selects continuing from the menu which is presented, all three sentences will be regenerated from the new selection expression, and displayed as in Figure 8.4.

Here the resulting French and German sentences are identical to the previous ones, because the

continuing/noncontinuing distinction is only required for English, as discussed in Section 8.1.

Figure 8.4: Multilingual generation (4)

Part III

Evaluation

Chapter 9

Evaluation of the demonstration prototype

The grammar embedded in the demonstration prototype is small in scope, but gives a fairly complete coverage of the generation of complex modal and auxiliary verb groups. The computational implementation is efficient, enabling the prototype to perform well as an interactive demonstration, especially in showing the ease and speed with which sentences can be revised. Examples of interactive revision were given in Section 7.2 for the monolingual version and in Section 8.2 for the multilingual version.

The response time of the system is so short that it takes the user just a few seconds to select a feature for revision from the feature summary table, select a new value for the feature from its system menu, select further values from dependent system menus when necessary, and then see the revised and regenerated sentence displayed. Since the grammar is so small, this speed cannot be taken as evidence for the efficiency of the implementation, or as an indication of possible response times of a large-scale system, but it means that the user gets rapid feedback on the realized effects of particular choices. This rapid feedback is important, because it provides a built-in way of learning about the choices and their realizations. If the user is presented with a menu where the meaning of a particular choice is not clear, it is possible to make an arbitrary choice, see the generated sentence, then go back and make the other choice and compare the result.

The ease and speed with which sentences can be revised suggests that this type of point-and-click

interface, directly manipulating the values of features used in generation, would be a good approach to knowledge-based post-editing in machine translation. This is discussed in Section 9.2.

9.1 Limitations of the implementation

The restriction of the scope of the grammar means that some issues, which would require successful solutions for a viable full-sized system following this approach, were not addressed.

Recursive embedding

Since only one clause is generated, with minimal noun phrases, a single table displaying the feature summary for the clause is sufficient for the purposes of the demonstration. In a full grammar, even with a single main clause, it would be necessary to display separate feature summaries for each noun phrase in the clause, to allow revision of determiners, singular/plural number, adjectival modifiers and so on. The noun phrases might include embedded relative clauses, requiring their own feature summaries, and these in turn would include further noun phrases, with no theoretical limit to the depth of recursive embedding. For full texts, higher levels of feature selections would need to be displayed for rhetorical structures, clause complexes, genre features, and so on.

A full solution to this problem would require an initial display of only the surface strings of the text, with a facility to present pop-up feature summary tables in response to mouse clicks on specific strings. This would require a method of linking the surface strings back to their feature summaries. It would also be necessary to provide a simple method for moving quickly up and down the levels of embedding of the summary tables, in order to reach the appropriate level at which the revision is desired.

Representation of realization rules

In a computational implementation of SFG, the representation of the realization rules needs to be absolutely clear. This is especially true when they include complex conditions, as in the Genesys form of rules used in the mini-grammar. There are various possible ways to organize the rules. One approach would be a procedural style, in which all the rules are strictly ordered, and all the conditions take the form "if (condition1) then if (condition2) then (action1) else (action2) endif else (action3) endif" and so on. Another approach would be a strictly declarative style, in which the rules are unordered, and all the

conditions are expressed as exhaustive logical conjunctions. Unfortunately, despite the extensive work on SFG-based generation systems surveyed in Part I, there is so far no really rigorous, perspicuous and widely established representation system for realization rules.

There is in fact a tendency in SFG literature for network diagrams to be presented very carefully and realization rules to be left relatively unclear. Fawcett himself criticises this tendency, and presents detailed realization rules for the mini-grammar in [Fawcett 1988], but the interpretation of the structure of the complex conditions in the rules is not immediately clear, and some effort was required to work out exactly what was required. In the later development of new sets of realization rules for French and German, even for such a small grammar, it was difficult to find the correct way to organize the rules.

In the implementation of the demonstration prototype, no attempt was made to develop a representation system for realization rules which would be rigorous and perspicuous enough for use in a large-scale system. It was easy in practice to produce Prolog code which implements the set of realization operations, and then to combine a semi-declarative form for individual realization rules with a particular ordering of the complete set of rules, following approximately the published rules, to achieve the working and bug-free demonstration system described in Chapter 6.

The example of agreement in French

As mentioned in Section 8.1, a linguistic problem arose in extending the monolingual system to the multilingual version. In the English-only version, since all the noun phrases in the mini-grammar are deliberately restricted to third-person singular proper names, subject-verb agreement is built-in by only including third-person singular forms of the finite verbs. In French passives, however, the passive participle must agree with the subject not only in number but also in gender.

The demonstration prototype therefore generates incorrect French passives when the subject is feminine. The example shown in Figure 8.2

"Fred avait été frappé par Phoebe"

(Fred had been thumped by Phoebe)

is correct because Fred is masculine, but the system will also generate

"Phoebe avait été frappé par Fred"

which is incorrect because Phoebe is feminine. The correct realization would be

"Phoebe avait été frappée par Fred"

(Phoebe had been thumped by Fred)

in which "frappée" takes the feminine ending.

It would be possible to add a new choice system to the *thing* network to choose male or female sex for the participant roles before choosing the name, with *male* as entry condition for the choice of men's names and *female* as entry condition for the choice of women's names. However, the mini-grammar on which the demonstration system is based provides no mechanism for handling subject-verb agreement. So a satisfactory solution would require the implementation of a new mechanism to handle agreement.

However, this problem leads into a set of more significant issues. Grammatical agreement, and the range of morphological variation involved in it, is an extremely restricted phenomenon in English in comparison with many other languages, including French and German. The emphasis in SFG on functionally significant choices, and the concentration on English (and in Halliday's case also on Chinese) has meant that other issues such as syntactic agreement, lexical gender and morphology have been relatively neglected. These issues are discussed further in Chapter 10.

9.2 An approach to knowledge-based post-editing

Most machine translation systems rely on post-editing by a target language expert. For export translation, as described in Section 4.1.2, post-editing is essential, and therefore the use of MT for export translation is restricted to professional translation agencies.

Professional post-editing

Although post-editing is such an essential part of the use of MT, and forms a bottleneck in the throughput of MT-based translation, not very much has been done to apply natural language processing techniques to improve the way the task is carried out. In fact, post-editing is normally done with just an ordinary word processor. This gives the post-editor the freedom to make totally unlimited changes, an advantage in the case of the professional translator (which is a totally different case from that of the monolingual user, discussed in Sections 4.2.1 and 4.2.2).

It is odd that, while sophisticated NLP techniques are used to produce the raw translation, and the MT system includes extensive grammatical knowledge of the target language, normally none of these resources are applied to helping the post-editor. In the raw translation the post-editor sees errors which could be described in SFG terms as incorrect choices in specific functional systems of the target language. The post-editor not only needs to perceive the errors at the *functional* level, but also has to manipulate the target text extensively at the level of strings and individual characters in order to *realize* the correct functional choice.

For example, in MT from Japanese to English, the absence of number features in Japanese nouns means that MT systems regularly make wrong decisions about singulars and plurals in English. When the post-editor sees that an English singular noun needs to be plural, it is easy to add "-s" or "-es" to its string with the word processor. However, changing the noun from singular to plural may also require a change in the determiner, a change in verb agreement, and changes in subsequent pronouns and other nouns. For example, suppose the raw translation is:

> "When the post-editor sees that an English singular noun needs to be plural, it is easy to add "-s" or "-es" to its string with the word processor."

If the post-editor decides that "noun" should be plural, it is necessary to make 5 different changes to produce:

> "When the post-editor sees that English singular nouns need to be plural, it is easy to add "-s" or "-es" to their strings with the word processor."

In the case of professional translation, it is reasonable to assume that the post-editor has sufficient target language knowledge to do all this correctly, as well as the subject domain knowledge that more than one thing is referred to by the noun. The MT system typically does not have the subject domain knowledge to make the correct functional choice for the number feature of the noun. However, the MT system does have the target language knowledge needed to realize the consequent agreement of the determiner and verb, and may be able to make a better attempt at the subsequent pronoun and noun.

The functional, point-and-click approach to revision in the demonstration prototype, described in Sections 7.2 and 8.2, suggests a new approach towards post-editing. Instead of changing the surface forms of the words, the post-editor can change the functional choices in the target language systems,

and leave the detailed realization of the change to the machine. It is necessary for the response time for the display of the regenerated sentence to be short enough for the post-editor to check that the desired changes have been made before moving on.

As mentioned in Section 9.1, for a full-scale system the displays of features would need to be much more sophisticated than in the case of the mini-grammar. However, the prototype system gives a good practical demonstration of what this approach could look like and how it could work.

Post-editing for monolinguals

Although improvements in post-editing facilities for professional translators are very desirable, the really significant challenge is to find ways to enable ordinary people to achieve successful personal export translation without professional post-editing. Some approaches to this challenge were described in Sections 4.2.1 and 4.2.2.

The approach taken in the demonstration prototype directly addresses this challenge. The *linguistic* knowledge of the target language is the responsibility of the system. The knowledge of the subject matter to be communicated is the responsibility of the user. The process of generating the target text is shared between the system and the user, as the system initiates inquiries when further information is necessary for linguistic decisions, and the user initiates revisions when the generated text is unsatisfactory. The user can stop worrying about grammatical correctness, which is guaranteed by the control imposed by the grammar, and can concentrate on the correctness of the choices required.

In terms of the three problems listed in the Introduction, following the framework of [Tsujii 1986], it is the third problem which is crucial. The knowledge of the target language functional systems (and their internal relationships) is built into the systemic grammar network. The knowledge of target language structures (and their relationships to the functional systems) is built into the realization rules. The hard part is how to help the monolingual user to understand and make those functional distinctions which are not made in the user's own language.

The prototype shows the basic approach of displaying felicity conditions, in Japanese, for the English choices. It would be best if the felicity conditions were initially provided by the grammar writers, and then tested and modified in practical use with Japanese users. In the development of the prototype, suggestions from Japanese colleagues were adopted in the felicity conditions, with the result that the

English and Japanese versions are not necessarily equivalent.

Further developments could provide more feedback to the user. It would be possible to add "help texts" with further explanations of the distinctions. It would be possible to add "back generation" from the set of choices into Japanese, to confirm the choices made. It would be possible to add "back translation" of the English surface text into Japanese, to check for misunderstandings. These further developments were outside the scope of the implemented prototype.

Chapter 10

Evaluation of Systemic Functional Grammar

10.1 Strengths and weaknesses of SFG

One of the strengths of SFG is that it addresses the need for a wide range of factors to be included in language processing systems, as described in the Introduction in the case of machine translation. The factors discussed by [Tsujii 1986] - semantic, pragmatic, discourse - are included in SFG as functional systems in the three metafunctions - ideational, interpersonal, textual - described in Section 1.2.3. Of course, there is nothing special about the number three, except that we feel easier thinking in three dimensions. In fact Fawcett distinguishes not three but eight functional areas of language - experiential, logical relationships, negativity, interactional, affective, modality, thematic (theme-rheme structure), informational (given-new structure) [Fawcett 1980].

A closely related but distinct strength is that these factors are not organized as separate strata at different levels, but are multidimensional. In SFG, there is a separation into two levels: the level of function and the level of form. Realization is the mapping between these two levels. However, at the functional level, all the metafunctions apply in parallel. This means, in non-SFG terminology, that semantics, pragmatics and discourse are on the same level. This contrasts with some other approaches in which syntax, semantics and pragmatics are considered to be three distinct levels.

Fawcett describes SFG as "holistic", and considers this as a strength:

> "...My point is that many of these matters are already incorporated into a HOLISTIC system in SFL - and, moreover, that many of the choices in the language system that have in some approaches been relegated to 'pragmatics' are in fact integral to the semantics and syntax of the clause..." [Fawcett 1988]

Morphology and syntactic agreement

However, SFG sometimes appears to push this holistic tendency too far. While it may be good to have semantics, pragmatics and discourse factors all operating in parallel on the same level - that of function - that level needs to be kept distinct from the lower level - that of form - at which syntax and morphology also operate in parallel. But there is a tendency in SFG for realization rules to be formulated in such a way that there is no scope for syntactic or morphological operations at the lower level. Realization rules often specify actual surface strings directly, rather than morphemes or lexemes.

For example, in the mini-grammar described in Section 5.2, [disposition & non-conditional & affirmative] is realized as "will", whereas [disposition & non-conditional & negative] is realized as "wo". This works in practice because in the mini-grammar the negative feature is always realized as the suffix "n't". However, if a further choice system were added, which resulted in the inclusion in the grammar of both "not" and "n't" as negative forms, then while it would be natural for the realization of the negative feature to distinguish the new factor in its conditions, it would also be necessary for the complex conditions on the realization of the disposition feature to be revised. This could be avoided if morphological variation could be handled at the level of form by morphological rules, rather than being part of the realization mapping from the level of function. Yet SFG does not recognize morphology as a level of language with its own rules.

An extreme case is the English indefinite article. The variation between "a" and "an" must be handled by phonological rules at the level of form. It would be absurd for realization rules to state that indefiniteness is realized as "a" if the complex condition includes the semantic feature "pearness", but as "an" if the complex condition includes the semantic feature "appleness".

Syntactic agreement in English is so restricted that it can be handled successfully on the basis of *semantic* features of number, animacy and sex ("natural" gender). In French and German, on the other hand, agreement depends on *lexical* features of "grammatical" gender. In an inquiry semantics approach to generation based on functional choices, it is appropriate to raise inquiries for semantic features such

as number, animacy and sex, but it makes no sense to treat grammatical gender in the same way.

However, the researchers on the KOMET project found that the techniques which the Penman project used successfully with the Nigel grammar of English did not provide elegant solutions to the problem of grammatical gender agreement in German. As described in Section 3.3, they solved the problem within the existing framework only by creating a new metafunction for agreement, and by arranging that the inquiries raised by the LEXICAL-GENDER-CHOOSER would receive responses from the lexicon. This appears to violate the idea that lexicogrammatical information is held inside the lexicogrammar, and the chooser/inquiry interface operates between the lexicogrammar and the environment in which it is embedded.

These problems led the KOMET researchers to consider the need to include some notion of head-daughter dependency in SFG. This idea is discussed in Section 10.2.2.

10.2 Scope for constructive exchanges

10.2.1 Semantic head-driven generation

SFG-based generation of a clause starts with a traversal of the clause network. Within the clause network, choices in the transitivity system of the ideational metafunction lead to realization of the main verb and preselections for some features (such as case) of the noun phrases which will fill the participant roles of the clause. The participant roles are then filled by secondary traversals of the noun group network. In this way, the generation is driven by the semantic head of the clause.

Working in a different framework, aiming at a uniform architecture for parsing and generation based on the development of declarative representations of grammars in information-based approaches such as UCG and HPSG, [Shieber *et al.* 1989] showed that semantic head-driven generation is more efficient than left-to-right generation.

> "The left-to-right scheduling of Earley parsing, geared as it is toward the structure of the string rather than that of its meaning, is inherently more appropriate for parsing than generation. This manifests itself in an overly high degree of nondeterminism in the generation process. For instance, various nondeterministic possibilities for generating a noun phrase (using different cases, say) might be entertained merely because the NP occurs before the verb which would more fully specify, and therefore limit, the options...
>
> Thus for generation, we want a traversal order geared to the premise of the generation problem, that is, to the semantic structure of the sentence. The new [semantic head-driven

> generation] algorithm is designed to reflect such a traversal strategy respecting the semantic structure of the string being generated, rather than the string itself." [Shieber *et al.* 1989]

This move towards semantic head-driven generation in work on information-based grammar formalisms brings these approaches closer to that of SFG. However, in SFG-based generation, the traversal of the clause network selects not only semantic features from the systems of the ideational metafunction, but also pragmatic features from the interpersonal metafunction and discourse features from the textual metafunction. All these features are included in the selection expression from which the surface string is generated by the realization rules.

The selection expression in SFG corresponds to the "semantic structure" referred to by Shieber as "the premise of the generation problem", that is, the logical form from which the surface string is generated by the rules of the grammar. However, the logical form in the information-based grammar formalisms with which he and others are working is typically restricted to a purely semantic structure. In contrast to the SFG selection expression, the logical form is therefore typically under-specified in interpersonal and textual information. The result of this under-specification is that several alternative surface strings, varying in interpersonal and textual factors, can be generated from the same logical form.

In response to this problem, rather than developing richer functional theories for the under-specified factors, researchers working within these formal approaches have concentrated on the efficiency of the generation algorithms. For example, [Haruno *et al.* 1993] adapt for generation the chart-based techniques developed in earlier work on the BUP parser, to avoid recomputation of partial results which can be shared among several alternatives. While efficiency of generation algorithms is extremely important, it seems that adoption of a wider range of factors, as in SFG, is more crucial to the problem.

10.2.2 HPSG

[Bateman & Momma 1992] contrast the strong points of SFG, oriented towards generation, and HPSG, oriented towards analysis, and propose that a combination of their respective strengths would be valuable:

> "The generation task has required extensive work on the more abstract strata: without a rich breadth of communicative goals, grammatical resources for expression cannot be satisfactorly utilized. ...

> In contrast to the generation perspective, work oriented towards analysis - particularly within current information-based grammars such as LFG and HPSG - has paid extensive attention to the less abstract strata of the linguistic system and has produced highly detailed accounts of syntagmatic organization. It naturally suggests itself, therefore, that a combination of the two paradigms should enable us to fill gaps in the respective coverage of strata offered by the two perspectives individually." [Bateman & Momma 1992]

The strengths of HPSG include elegant principles for handling head-daughter dependency relations and syntactic agreement. These are precisely the problems which the researchers on the KOMET project found difficult to handle in the SFG framework.

> "In computational systemic functional grammar (SFG) accounts of syntagmatic relations and their representation are generally less well worked out than treatments of paradigmatic relations, which have *per traditionem* been in the focus of theoretical interest in systemic linguistics. Leaving syntagmatic relations practically implicit presents a number of problems for the representation of certain linguistic phenomena, such as agreement, government, etc." [Teich 1992a]

In seeking to find a better solution to the problems of agreement than the one described in Section 3.3, the KOMET researchers wish

> "...to look at possible merits of a more explicit incorporation into SFG, and in particular into its computational instantiation, of the notion of linguistic dependency found in work such as HPSG, Daughter Dependency Grammar, and Word Grammar." [Teich 1992a]

Given that a combination of their respective strengths would certainly be valuable, it needs to be asked whether the similarities and differences between SFG and HPSG make this idea feasible.

As discussed in Section 10.1, the wide range of factors included in SFG operate multidimensionally in parallel rather than being organized into successive levels. HPSG similarly adopts a multidimensional approach. In HPSG, multidimensional lexical and phrasal signs include specifications of phonological, syntactic and semantic features. These features can share values and can be processed in parallel by unification. As mentioned in Section 10.2.1, the range of features tends to be relatively restricted compared with SFG, though HPSG does include both a "Content" field for semantic features and a "Context" field for pragmatic features.

However, there is a major difference between the lexicalist approach of HPSG and the approach of SFG:

> "Attempting such a "unification" of approaches might at first seem counter-intuitive because dependency grammars are highly lexically-centered whereas in systemic functional grammars linguistic information is not prominently tied to lexical heads but distributed over a number of units of different ranks. Implementations of SFG for natural language generation, such as PENMAN and KOMET, are based on a *situation-based* perspective whereas HPSG-style linguistic descriptions are *lexically-based*." [Teich 1992a]

10.2.3 Typed feature structures

A recent development in computer science is the elaboration of the theory of typed feature structures (TFS) [Carpenter 1992]. A TFS system was implemented at the ATR project [Emele & Zajac 1989], and subsequently applied to multilingual generation in Project Polygloss at the University of Stuttgart [Emele *et al.* 1990]. [Carpenter 1992] describes how both SFG system networks and HPSG feature structures and signs can be represented and manipulated in the TFS formalism.

[Bateman & Momma 1992] describe an experiment in which a small fragment of Penman was implemented in a TFS system. Part of the Upper Model (see Section 2.2.3), part of the system network, including realization rules, and part of the chooser/inquiry interface (see Section 2.2.2) were all translated into the TFS representation. After defining this fragment of Penman in TFS form, it was possible to run it in both directions, for generation and analysis. The analysis started with HPSG parsing to produce a minimum set of grammatical features needed to determine a unique semantic analysis.

The only sentence attempted was "Kim devours every cookie." Therefore this experiment was extremely inconclusive. However, further research may result in further progress in finding possible ways to combine some of the strong points of SFG with some of the strong points of HPSG by using the TFS formalism.

If such progress can be achieved, it could become possible to develop some kind of integrated HPSG/SFG parser capable of extracting a wider range of factors from source texts. Up to now, none of the experiments in parsing with SFG have had great success. However, if an HPSG/SFG parser could automatically extract a wider range of factors from source texts, and if further factors not present in source texts could be obtained by means of inquiries, as in SFG-based text generation, the requirements of [Tsujii 1986] for improving the quality of machine translation, discussed in the Introduction, would be much closer to being met.

Bibliography

[Bateman *et al.* 1991a] John A. Bateman, Elisabeth Maier, Elke Teich and Leo Wanner:

Towards an architecture for situated text generation.

International Conference on Current Issues in Computational Linguistics, Penang, 1991.

[Bateman *et al.* 1991b] John Bateman, Christian Matthiessen, Keizo Nanri and Licheng Zeng:

Multilingual text generation: an architecture based on functional typology.

International Conference on Current Issues in Computational Linguistics, Penang, 1991.

[Bateman & Momma 1992] John Bateman and Stefan Momma:

The nondirectional representation of Systemic Functional Grammars and semantics as Typed Feature Structures.

Proceedings of the 15th International Conference on Computational Linguistics, ACL, 1992.

[Bateman *et al.* 1993] John Bateman, Liesbeth Degand and Elke Teich:

Towards multilingual textuality: some experiences from multilingual text generation.

Preprints of Fourth European Workshop on Natural Language Generation, Pisa, 1993.

[Berry 1975] Margaret Berry:

An Introduction to Systemic Linguistics: 1 Structures and Systems, Batsford, 1975.

[Berry 1977] Margaret Berry:

An Introduction to Systemic Linguistics: 2 Levels and Links, Batsford, 1977.

[Carpenter 1992] Bob Carpenter:

The Logic of Typed Feature Structures, Cambridge, 1992.

[Dale *et al.* 1990] Robert Dale, Chris Mellish and Michael Zock (eds.):

Current Research in Natural Language Generation, Academic Press, 1990.

[Dale *et al.* 1992] R. Dale, E. Hovy, D. Rösner and O. Stock (eds.):

Aspects of Automated Natural Language Generation; Proceedings of the 6th International Workshop on Natural Language Genearation, Springer Verlag, 1992.

[Degand 1993] Liesbeth Degand:

Towards a systemic functional grammar of Dutch for multilingual text generation.

Preprints of Fourth European Workshop on Natural Language Generation, Pisa, 1993.

[Emele & Zajac 1989] Martin Emele and Rémi Zajac:

RETIF: A rewriting system for typed feature structures.

ATR Technical Report TR-I-0071, Kyoto, 1989.

[Emele *et al.* 1990] Martin Emele, Ulrich Heid, Stefan Momma and Rémi Zajac:

Organizing linguistic knowledge for multilingual generation.

Proceedings of the 13th International Conference on Computational Linguistics, ACL, 1990.

[Fawcett 1980] Robin Fawcett:

Cognitive Linguistics and Social Interaction, Julius Groos Verlag, 1980.

[Fawcett 1988] Robin Fawcett:

Language generation as choice in social interaction.

in [Zock & Sabah 1988].

[Fawcett 1992a] Robin Fawcett:

Felicity conditions and predetermination rules for the lexicogrammar: The generation of referring expressions as a test case.

Report No. 13 of the COMMUNAL Project, University of Wales College of Cardiff, 1992.

[Fawcett 1992b] Robin Fawcett:

Towards a view of the role of probabilities in generation.

Report No. 19 of the COMMUNAL Project, University of Wales College of Cardiff, 1992.

[Fawcett 1992c] Robin Fawcett:

Some issues in discourse planning: towards a marriage of the systemic flowchart model of exchange structure with rhetorical structure theory.

COMMUNAL Working Paper No. 3, University of Wales College of Cardiff, 1992.

[Firth 1957] J.R. Firth:

A synopsis of linguistic theory 1930-1955.

in F.R. Palmer (ed.) *Selected Papers of J.R. Firth 1952-1959*, Longman, 1968.

[Gazdar & Mellish 1989] Gerald Gazdar and Chris Mellish:

Natural Language Processing in Prolog, Addison Wesley, 1989.

[Halliday 1985] M.A.K. Halliday:

An Introduction to Functional Grammar, Arnold, 1985.

[Halliday & Hasan 1989] Michael A.K. Halliday and Ruqaiya Hasan:

Language, Context and Text: a social semiotic perspective, Oxford, 1989.

[Haruno *et al.* 1993] Masahiko Haruno, Yasuharu Den and Yuji Matsumoto:

Bidirectional Chart Generation Algorithm.

Preprints of Fourth European Workshop on Natural Language Generation, Pisa, 1993.

[Hovy 1988] Eduard H. Hovy:

Planning Coherent Multisentential Text.

Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics, ACL, 1988.

[Hudson 1971] R.A. Hudson:

English Complex Sentences, North-Holland, 1971.

[Jelinek *et al.* 1990] J. Jelinek, G. Wilcock, O. Nishida, T. Yoshimi, M.J.W. Bos, N. Tamura and H. Murakami:

Japanese-to-English Project PROTRAN & TWINTRAN.

Proceedings of the 13th International Conference on Computational Linguistics, ACL, 1990.

[Kameyama 1988] Megumi Kameyama:

Atomization in grammar sharing.

Proceedings of 26th Annual Meeting of the Association for Computational Linguistics, ACL, 1988.

[Kempen 1987] Gerard Kempen (ed.):

Natural Language Generation. New results in artificial intelligence, psychology and linguistics, Martinus Nijhoff, 1987.

[Kress 1976] Gunther Kress (ed.):

Halliday: System and Function in Language, Selected Papers edited by G.R. Kress, Oxford, 1976.

[Mann 1983] W.C. Mann:

Inquiry semantics: a functional semantics of natural language grammar.

First Annual Conference of the European Chapter of the Association for Computational Linguistics, ACL, 1983.

[Mann & Thompson 1988] W.C. Mann and S.A. Thompson:

Rhetorical Structure Theory: A theory of text organization.

in L. Polanyi (ed.) *The Structure of Discourse*, Ablex, 1988.

[Matthiessen 1987] Notes on the organization of the environment of a text generation grammar.

in [Kempen 1987].

[Matthiessen 1988] Christian Matthiessen:

What's in Nigel: Lexicogrammatical Cartography.

Unpublished USC/ISI Nigel grammar documentation, 1988.

[Matthiessen 1991] Christian Matthiessen:

Functional Grammar.

Unpublished materials for tutorial course at International Christian University, Tokyo, 1991.

[Matthiessen & Bateman 1991] Christian M.I.M. Matthiessen and John A. Bateman:

Text Generation and Systemic-Functional Linguistics - Experiences from English and Japanese, Pinter, 1991.

[Matthiessen & Halliday to appear] Christian Matthiessen and M.A.K. Halliday:

Systemic Functional Grammar.

in F. Peng and J. Ney (eds.) *Current approaches to syntax*, Benjamins & Whurr, to appear.

[Mellish 1988] Christopher S. Mellish:

Implementing systemic classification by unification.

Computational Linguistics, Vol 14/1, 1988.

[O'Donnell 1992] Michael O'Donnell:

A Contemporary History of Systemic Parsing.

Paper given at 19th International Systemic Functional Congress, Sydney, 1992.

[Sampson 1980] Geoffrey Sampson:

Schools of Linguistics, Stanford, 1980.

[Shieber *et al.* 1989] Stuart Shieber, Gertjan van Noord, Robert Moore and Fernando Pereira:

A semantic head-driven generation algorithm for unification-based formalisms.

Proceedings of 27th Annual Meeting of the Association for Computational Linguistics, ACL, 1989.

[Somers *et al.* 1990] Harold Somers, Jun-ichi Tsujii and Danny Jones:

Machine translation without a source text.

Proceedings of the 13th International Conference on Computational Linguistics, ACL, 1990.

[Steiner 1983] Erich Steiner:

Die Entwicklung des Britischen Kontextualismus, Julius Groos Verlag, 1983.

[Steiner *et al.* 1988] Erich Steiner, Paul Schmidt and Cornelia Zelinsky-Wibbelt (eds.):

From Syntax to Semantics: Insights from Machine Translation, Pinter, 1988.

[Teich 1992a] Elke Teich:

Dependency and systemic functional grammar.

Paper given at 19th International Systemic Functional Congress, Sydney, 1992.

[Teich 1992b] Elke Teich:

KOMET: Grammar documentation, Technical report, GMD/IPSI, Darmstadt, 1992.

[Tsujii 1986] Jun-ichi Tsujii:

Future directions of machine translation.

Proceedings of the 11th International Conference on Computational Linguistics, ACL, 1986.

[Tucker 1989] Gordon H. Tucker:

Natural language generation with a systemic functional grammar.

Laboratorio degli studi linguistici 1989/1, Universita degli Studi di Camerino, 1989.

[Vander Linden *et al.* 1992] Keith Vander Linden, Susanna Cumming and James Martin:

Using System Networks to Build Rhetorical Structures.

in [Dale *et al.* 1992]

[Wilcock 1991] G. Wilcock:

An experimental Japanese interface to a systemic grammar for generation of English auxiliary verbs.

Paper given at 18th International Systemic Congress, Tokyo, 1991.

[Winograd 1972] Terry Winograd:

Understanding Natural Language, Edinburgh, 1972.

[Wood & Chandler 1988] Mary McGee Wood and Brian Chandler:

Machine translation for monolinguals.

Proceedings of the 12th International Conference on Computational Linguistics, ACL, 1988.

[Zock & Sabah 1988] Michael Zock and Gerard Sabah (eds.):

Advances in Natural Language Generation, Volume 2, Pinter, 1988.