In J. Webster, R. Hasan, C. Matthiessen (eds), 2005. *Continuing Discourse on Language: A Functional Perspective*. Equinox: London. Pp 343-382.

SFL in Computational Contexts: a Contemporary History

Mick O'Donnell (WagSoft Systems)

John Bateman (Bremen University)

1 Introduction

Systemic Functional Linguistics (SFL) as we know it today is the result of a continual evolution of theory and description. The theory had its roots in Firth's teachings in linguistics, taken up and developed by Halliday, first in his system/structure description of Chinese (1950s), later in his development of Scale & Category Grammar (1961), and afterwards in the evolving complexity of SFL.

Intertwined with this evolution we find also the evolution of computers, and their use in relation to SFL. Computational applications of SFL, such as sentence generation and analysis, have enabled practitioners to explore just how complete the Systemic model of language is: when confronted with real texts, is the theory sufficient to provide formal instructions for interpreting those texts; and when confronted with meanings, is the theory sufficient for providing formal instructions that motivate natural texts corresponding to those meanings. In this way, the computer-using linguist (or computational linguists as they are now called), could find where the model failed to work, and needed to be changed. They have also found gaps in the theory: places where SFL does not tell us enough to allow us to generate a text, or to understand one, and thus point to where the theory needs to be extended.

This chapter summarises the evolution of computational applications which use SFL, or help those who use SFL. We look at those applications which use SFL for machine translation, parsing, text generation and dialogue systems. We also look at tools which help the practitioner: coding tools and system network drawing tools.

2 Machine Translation

During the 1940s, the first modern computers[i] were created. They were extremely large, sometimes the size of small buildings. The first computers were programmed by setting switches on a panel. Gradually the computers became smaller, less expensive, and easier to program. Time on these computers became available for a wider range of uses. Some considered that the raw power of these computers could be turned to machine translation (MT) between languages. During the 1950s in Britain, two projects in this direction were started, one involving Firth's group in University College London, the other involving a group in Cambridge which included Halliday.

2.1 Firth & Booth

During 1955, Andrew D. Booth's group at Birkbeck laboratory, London was funded by the Nuffield Foundation to develop "a modest programme ... to render scientific French into acceptable English" (Booth 1956). Booth selected French in order to "avoid competition with American projects" and also because French was seen as an

easy starting point, since English and French word order were supposedly relatively similar (Hutchins 1986).

To develop the grammar and lexicon for the project, Booth contacted J.R. Firth at the University of London. Firth gathered a team of those in his department to develop the resources, which were a "microglossary" (less than 1000 words) and a "micro-grammar".

The approach of the project was to perform word-for-word translation, followed by some basic phrasal re-organisations. A sample output from the system (from Hutchins 1986) is as follows:

```
We demonstrate in this chapter the fundamental theorem of M.
Poincaré, after having studied the integrals of a system of
differential equations, considered as functions of initial values
```

It seems that the system worked on this limited lexicon, but the machine was too limited in storage capacity to enable real testing.

2.2 Halliday & the CLRU

In 1956, the *Cambridge Language Research Unit* (CLRU), of which Halliday was a member, received funding from the U.S. National Science Foundation to explore MT. The group also included Margaret Masterman (the director), Arthur Frederick Parker-Rhodes (a mathematician) and Dick Richens (a biologist). The central idea was from Richens: to develop an inter-lingua, such that each (European) language would first be translated into the inter-lingua, and thence into the target language. According to Wilks (1995), the inter-lingua, called NUDE (from 'naked ideas'), involved semantic categories such as FOLK, STUFF, MAN, THING, along with brackets and connectives.

It is not clear how much of the project's direction came from Halliday, although one commentator (Hutchins 1986) attributes some of the research to Halliday, citing Halliday (1956). Halliday commented on his involvement:

> "I felt that machine translation had an important political role to play. There were lots of cultures around the world where people were beginning to be educated in a mother tongue and if you could possibly have a machine to translate a lot of text books at least it would help the process." (Kress *et al* 1992).

Given that this project involved working with a hierarchical organisation of semantic categories, it seemed likely that Halliday might have been influenced by this project in the development of system networks. However, when asked, he said that the idea of system networks rather sprang from "taking seriously the paradigmatic project that was implicit in the Firthian (and also Helmslevian) interpretation of post-Saussurian linguistics." (personal communication). He was never able to convince the others on the project that they should give as much priority to system as to structural descriptions.

While no working MT system arose out of the project, the NUDE project had fairly major influence on the field in later years. Hutchins (1986), in summarising the history of MT, said that:

> "In recent years, research in Artificial Intelligence has turned increasingly to the areas of investigation which were first examined in depth by the Cambridge project ... features of the CLRU conception of semantic message structure have

lived on in various guises in both AI and MT research" (Hutchins 1986: section 5.3).

2.3 Later Work

In 1966, a U.S. government report concluded that automatic machine translation was not currently feasible. As a consequence, all U.S. government funding of MT stopped, and work in MT was greatly reduced even in Europe and the USSR. Research into MT only began to pick up again in the 1980s, due to increased linguistic sophistication, more precise understandings of the practical niches for which MT could provide solutions, and more powerful computers.

The involvement of SFL in MT remained largely indirect however. In the late 1980s, Erich Steiner constructed a model of clause types and participant roles for the German component of the large-scale European translation project Eurotra (Steiner *et al.* 1988). Steiner also subsequently interacted with the Natural Language group at the Information Sciences Institute (ISI) in Los Angeles (home of the Penman project: see below) in their first moves towards considering MT (Bateman *et al.* 1989). In the early 1990s, Graham Wilcock investigated the possible advantages of using SFG in machine translation, although only part of an MT system was produced (Wilcock 1993). And, in the mid-1990s, the Natural Language group at ISI then became one of the founding members of the large Pangloss MT project. This project was based on the idea of an interlingua, as espoused in the NUDE project. This time, the interlingua was based on a systemically inspired experiential semantics, the Penman Upper Model, that we will return to below. From an interlingua specification of an utterance, the Penman generation system was initially used to generate sentences into English, although this was later replaced by more statistically-based methods.

Subsequent SFL applications for MT have been, and continue to be, seriously hampered by the difficulties of achieving full-scale Systemic analysis components, to which we now turn.

3 SFL & Text Analysis

After the death of MT in the Sixties, attention instead turned to syntactic analysis of text. Syntactic analysis, or parsing, involves the interpretation of text in terms of the syntactic structures described in a grammar.

3.1 Parker-Rhodes & Yorick Wilks

The earliest attempt to parse with a Hallidayan grammar was during 1962-3. Parker-Rhodes, who worked with Halliday on the NUDE project, developed a parser for a grammar based on Halliday's Scale & Category grammar. Directing a crew which included Yorick Wilks, subsequently a highly influential figure in computational linguistics, he first of all developed "a system of syntax which was Hallidayan, with a lattice theory imposed on top to model the inclusion of grammatical types" (Yorick Wilks, personal communication). A lexicon was developed, outlining the possible governors of each word. Parker-Rhodes developed a method for parsing the grammar, which Wilks then programmed on a Hollerith punch card machine. The output of the program was a bracketing of the constituency of the input sentences, for example:

```
((His second wife) (was(young and (very beautiful))))
((His eyes) (were blue(like(the morning sky))))
```

3.2 Winograd

Terry Winograd, who later became one of the most influential researchers in Artificial Intelligence, was the first to use SFL in a parsing system. From the United States, he used a Fulbright scholarship to study for a year in Halliday's department in London, attending lectures under Halliday, Hudson and Dixon. While there, he did some interesting work on the computer analysis of tonal harmony using the Systemic formalism. After this, he returned to the States, taking up his enrolment at MIT.

Although at MIT, he was not in Chomsky's department, where his new Systemic approach to language would not have been appreciated. Rather, he was in the Artificial Intelligence Lab, under one of the originators of AI, Marvin Minsky. Minsky and Chomsky were actively hostile to each other, so when Winograd proposed using SFG for his doctorate project, there was no complaint. Winograd comments:

> "My direct source of inspiration came from the year in London where I studied with Halliday. If you look at what Halliday was doing in the context of the larger picture of generative grammar, he was trying to adapt his theories, which came more from a social perspective, to a generative form." (Thanning Vendelø1 2002: 10-11)

Winograd developed a system which allowed a human to communicate (by keyboard) in restricted English with a virtual 'robot' (represented on the computer screen). The robot's universe consisted of a set of blocks of different shapes and colours. The human could ask the robot questions about the current state of blocks ('which block is under the brown one?') or give orders, moving blocks about. The syntactic analyser of the system was based on Halliday's early SFG, centred around system networks for clause, group and word.

The system had very high impact at the time. The focus in the early 1970s was on the parsing of problem sentences, or on making syntactic parsers more efficient. Winograd, following a more functional line, dealt with the core sentences of the language, and focused on making the system work, not proving that it should work in theory. Where the current work was focussing on syntax, he made a system which analysed input text, interpreted it semantically, decided on the appropriate response to the input, and then acted upon it. The system required work in knowledge representation (modelling the blocks world) and on problem-solving. Many considered the system to have demonstrated that natural language interaction with artificial intelligences was possible, if not solved.

The success of this work spread the name of SFG in the NLP community. However, the success of the system as a functioning SFL parser needs to be balanced against several properties which subsequent computational work has shown to be problematic:

- *System network only used implicitly*: although Winograd used a system network, "the network itself was not input into the computer. The network is not referred to by [the system] but only by the person who is writing the grammar" (McCord 1977: 257)
- *Procedurally implemented*: syntax and semantics were procedural: directly coded into the program rather than being 'declaratively' stated. This is an instance of a general trend in computational linguistics: where the first work in an area is implemented procedurally, later work takes advantage of the

procedural model and produces a declarative version. The difference between these can be understood by imagining two devices: the first a bread-making machine, the second a general purpose cooking machine that takes recipes as input. The first has the recipe for bread encoded 'procedurally' in the concrete operations that it can perform; it may be very good at making bread but will not make good lasagna. The second takes an abstract declarative set of instructions, the recipe, and follows those instructions to produce bread or lasagna or anything else. In the case of computational linguistics, the recipes may correspond to grammars for different languages, or for different registers, etc. The advantages of the more general purpose approach should be clear.

- *Limited function structure*: function structure seems to be limited to one layer of function structure at each rank.
- *Domain-specific*: Winograd's grammar was tailored to the domain of BlocksWorld. It was quite small in coverage (120 features or so), and would have been difficult to increase substantially in size.

3.3 Michael McCord

Inspired by Winograd's work, Michael McCord started exploring SFG, soon found Dick Hudson's book, "English Complex Sentences" (Hudson 1971), and started up a correspondence with him. He spent the fall of 1977 with Hudson on a sabbatical.

He developed a modified version of SFG (McCord 1975), for which he then implemented a parser (McCord 1977). The main differences from standard SFG were:

- *All systems were binary in nature*.
- *Realisation rules were somewhat transformational*: the could add, move or delete functions.

In implementing his parser, he made an additional change: the grammar was re-represented in a form closer to the surface structure of clauses.

> "The structure rule for a base feature like CLAUSE is a single linear list of *segments*. Each segment consists of a *slot*, and a sequence of conditions and actions for *filling* the slot. ... Each slot is an atomic symbol such as SUBJECT or MAINVERB representing a grammatical function of the item that fills the slot. ... A structure rule expands a base feature into a *single linear list*, in overall form like a single phrase structure rule." (McCord 1977: 258)

Parsing a clause involves simply taking the structure rule for the clause, and testing for the presence or absence of each slot in the rule in turn.

In contrast to Winograd's system, this system represents the system network internally, and it is used to test feature inheritance, etc. The grammar allows for conflation, which means multiple layers of function structure can be handled, although it is not clear if this is true of the parser as well.

McCord's next system was called "Slot Grammar" (McCord 1980), which moved further away from its Systemic basis, yet still was partially derivative. Since 1983, McCord has been working for IBM, developing Slot Grammars for several European languages. These grammars are used within the IBM WebSphere Translation Server, reputedly one of the best translation systems available today.

3.4 Martin Kay

Kay started working on language at the CLRU several years after Halliday had left. Kay however visited Halliday in London several times in the 1960s, and in the 1970s, while Kay was working at Rand Corp (Los Angeles), Halliday visited him there. Kay started to work on Systemic Grammar, and this task led him to develop a formalism called Functional Unification Grammar (FUG), a forerunner of one the most widespread linguistic formalisms used today. He says:

> "FUG was in large measure an attempt to provide a formalization of Systemics that would be amenable to computation. In particular, Halliday spoke of 'realization rules' that would effectively translate the choices made in his networks into strings, but there was virtually nothing said about what these would be like, and I was eager to fill that gap." (Kay, personal communication)

While FUG has clear debts to SFG, it differs in substantial ways. Firstly, it does not include a separate paradigmatic component: there are no system networks (although features are used in the grammar, they are not organised in terms of systems).

A FUG is then, in a sense, a generalisation over the structures generated by a SFG. A Hallidayan grammar uses distinct graphical representations for structural constraints in the network and their corresponding occurrences as instances, as shown in figure 1. One step Kay took was to choose to work with only one notation for both grammar and analyses. His notation follows more closely the organisation of tree diagrams, but since such diagrams are difficult to type, he developed the notation shown in figure 2.

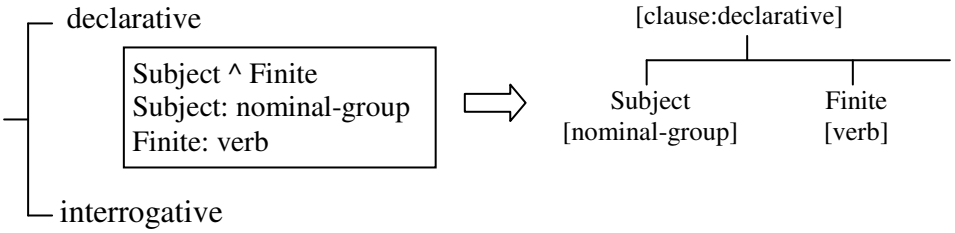

Figure 1: Systemics uses distinct representation in system networks and structure diagrams

```
[ CAT     = CLAUSE
  SUBJECT = [CAT = NOMINAL-GROUP]
  FINITE  = [CAT = VERB]
  MOOD    = DECLARATIVE
  {SUBJECT FINITE} ]
```

Figure 2: FUG representation

The key notion in Kay's grammar was one of *unification*: the merging together of distinct but compatible functional descriptions. Unification has always been implicit in SFG, via the use of simultaneous systems which need to be considered together to construct the structure and, to a certain extent, via some aspects of conflation. Kay made this notion explicit.

In the 1980s, Kay developed a parser for FUG (Kay 1985). While Winograd and McCord implemented their grammars procedurally, Kay *declarativised* the grammar (represented it in an implementation-independent form) and his parser accessed the

grammar as data. The parser was therefore able to process *any* grammar provided in the appropriate form, a clear advantage over previous procedural accounts.

He also introduced the notion of *compilation* of the grammar: automatic re-representation of the grammar in a form more suitable for a particular application. The canonical form of the grammar was used for generation, but needed to be represented in an alternative form for efficient parsing.

Kay himself was not happy with the parsing results and abandoned the formalism (personal communication). The formalism caught on however and is widely used in computational linguistics, usually extended to include something akin to system networks (e.g., HPSG, Typed Feature Structures) as we will discuss below.

3.5 Cummings & Regina

In the early 1980s, Michael Cummings in Toronto was working on the analysis of Old English nominal groups (NGs). Initially hand-drawing analyses on cards, he soon realized he needed to computerise the study to properly utilise the data. He recruited a programmer, Al Regina, and together they developed a system which allowed them to enter the syntactic analyses of the NGs, and ask questions of the database.

Because it was time-consuming to enter each analysis, they extended the system to parse each entered NG. The system did not require the specification of a grammar or lexicon, but rather derived such from the existing corpus. Each analysis was stripped of the lexemes, and stored away as a possible NG structure. A lexicon was constructed by examining the various lexical features that each occurring token was assigned.

This is, we believe, the first instance of TreeBank parsing (e.g., Charniak 1996), a method which became popular in the 1990s and is still widely used. One limit of the Cummings-Regina approach is that it does not attempt to generalise: the corpus may contain “seven red cats” and “very red cats”, but will fail to recognise “seven very red cats”, as only full structures are taken into account. Later approaches attempt to derive structure rules from the corpus, which capture such generalisations. Also, problems would arise moving away from the group to the clause, as clauses offer far more variety in their possible forms.

3.6 Robert Kasper – Legin

In 1985, Bob Kasper started working at the Information Sciences Institute (ISI) in Los Angeles on the Penman generation project (see below). His task was to see if the NIGEL grammar, then the largest generation grammar in existence, could also be applied to parsing. Because he was more familiar with Kay’s FUG, and FUG had already been applied to parsing, he chose to re-represent NIGEL into FUG.

Faced with overwhelming complexity, Kasper’s system (Kasper 1988) firstly parsed each sentence with a small (hand-written) phrase-structure grammar (PSG) (containing around 60 rules). Each PS rule also contained information for mapping the phrase structure of the rule onto a parallel systemic structure tree (see figure 3).

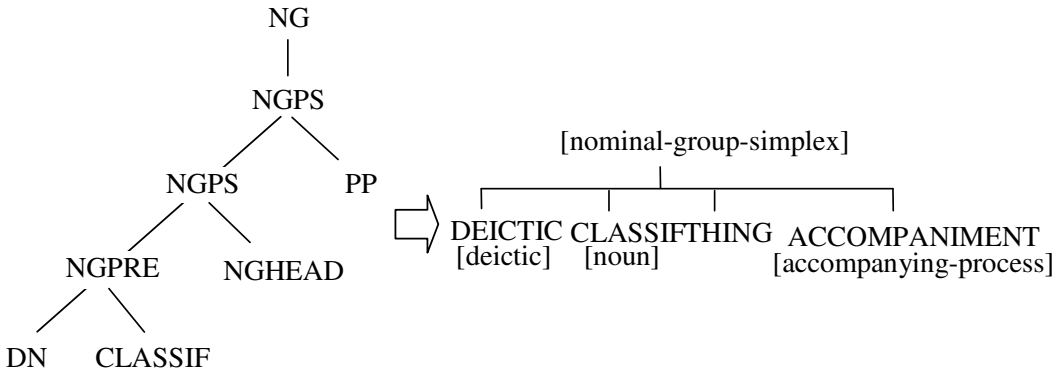

Figure 3: Mapping from context-free backbone to Systemic analysis

Once all possible analyses were produced (using a bottom-up chart parser), each alternative Systemic analysis was further enriched using information from the NIGEL grammar. This approach to parsing is often referred to as parsing with a "context-free backbone", as the PSG offers a low-detail 'skeletal' analysis, fleshed out by the functional grammar.

This was the first attempt to parse with a grammar in the full Hallidayan formalism, and of large size. However, the dependence on the PSG grammar lessens the importance of the work in the current context. The parser was quite slow, and was limited to parsing those structures covered by the PSG.

Later, with Mick O'Donnell and Richard Whitney, the work was repeated using a knowledge representation system, LOOM. They declarativised the (procedurally implemented) semantic mapping interface of Penman, thus allowing for an ideational layer analysis to be derived for each sentence, although the system only worked for very short sentences due to combinatorial complexity. This was, however, the first effort to produce a semantic stratum analysis using SFG.

3.7 Cardiff Approaches

During the 1990s, there were a series of parsing experiments based around Robin Fawcett's Systemic model. Fawcett had realized very early that the ability to ascribe Systemic analyses to texts automatically would be of tremendous significance if it could be made to work. In the Hallidayan model, there is system and structure at every level of linguistic representation. Thus, we have a system network for the grammar, which describes the grammatical structures at that stratum and, equally, for those approaches with a semantics, there is a distinct semantic network, which builds in turn semantic structures. In the Fawcett model, however, there is one system network, called a semantic network. The realisations of features in this network directly build syntactic structures. Fawcett thus uses a single network to model what the Hallidayan approach models over two strata.

In Halliday's grammatical structure, each element is described by a selection expression from the network (a set of features), and by the set of functions it fills in relation to its parent unit. The Fawcett approach is similar in this regard, but in addition, each element of structure is assigned a single syntactic 'class' (*Cl* for clause, *ngp* for nominal-group, etc. These classes are not part of the network). Each element gets exactly one class. The function structure and the syntactic classes are considered the syntactic structure, while the selection expression is considered the semantics of each unit.

Formally, i.e., in terms of the computational task that is to be solved in each case, what Fawcett calls semantic analysis is equivalent to grammatical analysis in the Hallidayan approach – the analysis of a sentence using a system network and its realisation rules.

3.7.1 Corpus-based approaches

The first efforts at parsing Fawcett-style grammars took a similar approach to Cummings & Regina: they ignored the grammar, instead extracting a (non-systemic) grammar and lexicon from a parsed corpus. These resources were then used for parsing.

Tim O'Donoghue took a corpus produced using Fawcett & Tucker's Genesys system to generate 100,000 sentences at random (all syntactically correct but with no regard to meaning). From this a grammar was extracted:

> "Rather than considering the horizontal slices of the tree – the phrases – we consider vertical slices. These vertical slices are called **strips**, hence the name VSG. A strip is a vertical path of nodes in the tree starting at the room and ending at a lexical node" (O'Donoghue 1991a: 4)

A compilation step extracts out of the parse corpus all possible strips. Another phase derives a 'collocation' network which shows which strips can follow a particular strip. This phase also identifies which strips can start and end a sentence. Another compilation produces the lexicon, which is a mapping from each word to the set of strips that that word can be the terminal of.

To parse, one simply finds the strips for each lexeme in the sentence. The strips for the first word which cannot begin a sentence are eliminated, as are those strips of the last word which cannot end a sentence. A valid parse select one and only one strip from each of the strip sets. The strips need to be compatible with each other (they must define a valid syntactic structure) and collocation rules need to link each strip to the next. The parser finds all possible combinations of the strips meeting these criteria, and these are output as possible parses.

The parser was only tested on further sentences generated from the grammar, so it is not clear how well the system would work on a real corpus. There are also potential problems of combinatorial explosions in the step which looks for possible combinations of strips, if a larger corpus is used.

O'Donoghue (1991b) added a further level of analysis: given a syntactic analysis of a sentence, the SFG is used to discover the set of Systemic features (for each unit) which would have produced the analysis.

Clive Souter used a hand-analysed corpus, a 65,000 word corpus of children's spoken English. Like O'Donoghue's system, Souter's used horizontal collocation information and vertical strips. However, Souter's vertical strips were limited to three levels of structure (vertical trigrams), enough to provide context when determining which daughters can attach to which parent units (Souter 1996).

3.7.2 A Grammar-based approach

In contrast, Ruvan Weerasinghe used Fawcett's grammar as the starting point (Weerasinghe & Fawcett 1993). He rewrote (by hand) the SFG in a form more amenable for parsing. In particular, two types of rules were produced, and used in a bottom-up chart parser:

- *Phrase structure rules*: rules stating the potential sequence of functions for each basic class of item (clause, nominal-group, etc.).
- *Function potential statements*: for each basic class, a statement of the set of functional slots that the element could fill (e.g., that a nominal group could fill the Subject slot, Complement slot, etc.). Similar statements are provided for words as well as basic classes.

We believe that the production of phrase structure rules for Systemic clause structure is only possible when limited grammatical possibilities are considered. Fawcett's grammar was written for generation, and thus only needed to account for a subset of possible expressions. In parsing, all possible orderings of elements need to be allowed for. For a grammar with real coverage, such simple structure rules are not sufficient.

Conflation was very limited in the system, thus avoiding the second major cause of complexity in Systemic parsing -- functional layering.

3.8 O'Donnell

In 1989, Mick O'Donnell experimented with several parsers for small Systemic grammars, but found difficulty when scaling up to larger grammars. In 1990 he worked at ISI under Bob Kasper, where he learnt about grammar recompilation, use of formal logic systems for parsing, and methods for handling complexity. In 1991, he joined the EDA Project[ii], funded by Fujitsu, with the project goal to parse translated computer manuals and critique the discourse structures. He developed a parser which worked directly on SFG, using a subset of the Nigel grammar. The parser recompiled the SFG into two resources: the set of possible function bundles allowed by the grammar (along with the bundles preselections) and a resource detailing which functions can follow a particular function (see Matthiessen *et al* 1991; O'Donnell 1993; O'Donnell 1995).

This was the first system which allowed parsing directly from a reasonable scale SFG, without need of additional specifications or a phase-structure backbone. However, when attempting to handle all of the NIGEL grammar, the system would take hours to compile the grammar, making grammar development difficult. As a first step towards handling this difficulty, O'Donnell he developed his own SFG, one more suited to parsing (using only the Mood layer). This smaller grammar could be precompiled quickly, and allowed quicker parsing of sentences.

In 2001, O'Donnell joined a Belgian Informatics company, where he lead a team to develop a Systemic-oriented parser. He had reached the conclusion that dependency grammars allowed faster parsing. Together with Joeri Van Der Vloet and Frederik Coppens, they developed a wide-coverage grammar & lexicon, based on the Systemic formalism (written in terms of a system network). One simplification of this approach was that each element connects to another via one function only, avoiding the complexity introduced by conflation. Also, ordering was done in terms of slot positions relative to the head, rather than the relative positioning rules of standard SFG. The system relied heavily on statistics to choose the best parse among those produced for a sentence.

This parser was able to parse previously unseen texts, with 92% of all words connected to the correct head-word via the correct relation.

O'Donnell is currently developing another parser for constituency-style Systemic grammars, assuming one function per unit, and using slot-based ordering rules.

3.9 Bateman et al.

John Bateman, interested in the formal properties of SFG, explored the representation of SFG in a formally established and well understood representation system: the Typed Feature Structures (TFS) system, descended from Kay's FUG mentioned above. Together with Martin Emele, Rémi Zajac and Stefan Momma of Stuttgart University they mapped a fragment of the NIGEL grammar and semantics into TFS (Bateman *et al.* 1992). Only those resources needed to parse a handful of variations around the clause *Kim ate every cookie* were included. Within TFS, they could assert partial syntactic representation of a sentence, and the system would expand this information using the grammar to build all the implied structure. No actual parser was implemented for the grammar, but later, they used an HPSG parser to produces a skeletal function structure for the sentence, which was then fleshed out by the Systemic resources in a way similar to the use of a skeletal phrase structure grammar described above (cf. figure 3).

The experiment was interesting in regards to the bidirectional use of SFG, the re-expression of SFG into the TFS formalism, and the hybridisation of HSPG syntax with SFG semantics. But, as a consequence, the system was not then a true SFG parser. Subsequent work exploring a pure SFG representation of large-scale grammars took this further, but explored formalisation issues rather than practical analysis; we therefore return to this below.

3.10 Summary

The main problem of using SFGs for parsing is that they are far more complex than post-Chomskian grammars: the grammar is closer to semantics than to form, system networks represent large numbers of simultaneous combinations of features, and the grammar involves multiple layers of function structure which need to be conflated together.

Some of the approaches avoid parsing with SFG entirely: they parse with a cover grammar, using the SFG to flesh out the description, while others ignore the grammar, instead using a corpus annotated with the grammar as a starting point.

Other approaches do use SFG for parsing, but limit the problem in various ways: assuming only a single layer of structure, or restricting themselves to a low-coverage grammar. Others aim for high coverage, but reduce the depth of information provided (e.g., using only Mood analysis).

We are still waiting for a parser which can parse in terms of a large Hallidayan style grammar. Meanwhile, we are getting good results with reduced formalisms, producing at least one system which is in commercial use (O'Donnell's last parser).

4 SFL & Text Generation

The other main branch of natural language processing is called *natural language generation* (NLG); this is often talked of as the 'opposite' of analysis or parsing, although there are many asymmetries between the two, both in terms of approaches and methods and in terms of research communities. Simply stated, NLG is the computational process of automatically producing sentences in some human language on the basis of a specification of communicative intention. A program that perform this task is called a *sentence generator*. From the perspective of SFL this can be viewed most usefully in terms of stratification. NLG is a 'stratification-descending'

process by which specifications, typically given at the computational equivalents of field, discourse semantics, or experiential semantics, are 'passed through' a lexicogrammar in order to produce, fully automatically, corresponding 'wordings'.

It is in NLG that some of the most close and productive exchanges between SFL and computational linguistics have occurred--generally because, as we shall see, it is the generation perspective on linguistic resources that fits most naturally and easily with the SFL-focus on *systemic potential*. In many respects, a systemic network as found within a lexicogrammar *is* a statement of how to construct a sentence or other grammatical unit. Following the network through, from left to right, invoking realization statements as they are encountered, is a recipe for constructing corresponding grammatical units. Thus, despite the fact that the non-computational user of systemic theory will tend to think first of analysis, within the computational context it is not analysis that the SFL framework has shown itself to be most suited to. This result, which has actually only become clear because of the attempts to apply SFL theory for both analysis and generation, has some fairly deep consequences for theory and for how systemic potential can be best captured. We will return to some of these consequences in our discussions below.

The first attempt to employ a systemic specification for automatic generation was Henrici (1965). This took the notions of systemic network (paradigmatic organization) and realization statements constraining structure (syntagmatic organization), and specified them sufficiently explicitly for a computer programme to carry them out. The importance of this part of the exercise--making a specification sufficiently explicit as to be computationally executable--cannot be over-emphasized and forms the heart of computational linguistics. The degree of explicitness required for computational specification is unforgiving; nothing can be left to the charitable interpretation of a human analyst. Although this may be less appropriate in areas of investigation where we just do not know enough to be precise, or in areas where the very applicability of precision is doubted, for tasks such as providing a tight and precise account of lexicogrammar such computational precision can be very rewarding. If the realization statements that one has written in a systemic network at two places in that network, perhaps quite distant but paradigmatically coherent, result in incompatibility when selected, then the grammar is wrong: this is not a matter of interpretation; most usually it is simply an error on the part of the grammar writer. And such errors occur with dramatically increasing frequency once the size of the resource being constructed, for example, the lexicogrammar for a language, increases to that necessary for detailed description.

Finding such unwanted incompatibilities and forcing the grammar writer to confront and resolve them is then perhaps the most important contribution of pursuing a computational instantiation of a linguistic theory. All of these aspects became explicit with computational applications of the systemic account for generation. Henrici's approach, although naturally limited in many respects, was the first step in showing that the computational application of systemic grammars for generation purposes was in principle possible.

Another significant move towards applying SFL for NLG was Robin Fawcett's (1973) consideration of what would be necessary to make a systemic account of grammar sufficiently explicit to drive a computational instantiation. Fawcett's work led subsequently to one of the longest running and extensive computational explorations of systemic linguistics for NLG and one of the most extensive

computational grammars: the COMMUNAL system developed in Cardiff throughout the 90s (Fawcett & Tucker 1992).

The most influential forerunner of the broader take-up of SFL within NLG occurred in the 1980s however. This was the PROTEUS system, written as part of the doctoral dissertation of Anthony Davey in Edinburgh in 1974 (published as Davey 1978). Here for the first time we find not isolated generation of sentences, but the use of a systemic lexicogrammar for the automatic production of full texts. These texts were naturally rather simple. They were running commentaries of games of noughts and cross ('tic-tac-toe') that were also played automatically by the computer. Nevertheless, they needed to display awareness of issues of cohesion, appropriate pronominalization, clause connectives as well as managing to get the basic grammar of the clauses produced correct. Davey's system was a convincing demonstration that it was feasible to produce extended texts automatically on the basis of machine-internal non-linguistic information; recent NLG systems generally follow in this tradition, producing information from database contents, from knowledge bases of expert systems, and from user-provided abstract input.

The role of SFL within NLG changed dramatically with the advent of the Penman text generation system, originally designed and conceived by William C. Mann of the Information Sciences Institute in Los Angeles. Mann was planning a large-scale NLG project, but found in an extensive study of the technology of the time that the tools available showed little linguistic sophistication. On the basis of previous computational experiences, Mann had been convinced that making a computer 'function as an author' (Mann & Moore 1980) would require detailed and linguistically valid models of lexis, grammar, semantics and discourse in order to succeed. While this does not sound surprising from the perspective of linguists, such an extensive incorporation of independently developed linguistic insights within computational models was far rarer within the NLG and dialogue systems that had been pursued hitherto and Mann's insight led directly to a close interaction between systemic theory and computation. Partly on the basis of results such as Davey's, Mann decided that what his project required was a computational instantiation of Michael Halliday's systemic-functional grammar as the 'front-end' of a larger, general-purpose NLG system. This system, Penman, was intended to be a register-non-specific, broad-coverage generation system, capable of producing texts in natural English for as broad a range of domains as required. To achieve this aim, he involved Halliday, and a fresh graduate student at UCLA working on a systemic interpretation of tense in English, Christian Matthiessen. Together they started on the computational specification and implementation of the extant fragments of English grammar within the systemic-functional framework that Halliday had been working on over the previous 15 years. The result of these efforts was the Nigel grammar of English, a fully explicit computational systemic-functional grammar for sentence generation that, by the late 1980s, had become the most extensive such computational grammar in any framework (Mann & Matthiessen 1985; Matthiessen 1985). Even today, there are few grammars with comparable coverage and the Nigel grammar continues to be used and developed; many have contributed to the grammatical descriptions it contains.

The approach to theory adopted within the Penman project adopts stratification as a strong organizing force. The component that is most familiar to systemicists is the grammar. This is the main hub, or engine, of the entire generation system. All of grammar is modelled as a single network, the first system of choice being the RANK

system. Further subnetworks then specify the grammatical possibilities for each rank, initially: clauses, groups and phrases. The Nigel grammar of English, written in the style defined by Penman, contains around 700 grammatical systems and 1500 grammatical features.

While a resource the size of Nigel is a substantial linguistic achievement, it still requires further components to be of use for automatic NLG. Indeed, the problem at hand is a general one facing all uses of systemic-functional resources for generating texts and sentences and several different systems have approached it in varying ways. In order to make use of a computationally specified systemic network representing the functional potential of a language, it is also necessary to *motivate the actual choices of grammatical features* that are to be made within concrete texts. Since a simple clause would generally involve 60-80 'choices' at clause-rank alone, this cannot be done sensibly 'by hand'. It is not enough for a sentence generator to specify that the language offers a minimal functional alternative between, for example, indicative (simple statements) and imperative clauses (simple orders), which in turn bring specific constraints to bear on structure; it is *also* necessary to specify the precise conditions under which indicative must be chosen over imperative, or vice versa. The functional motivation and orientation commonly assumed to underlie the system network organization must therefore also be spelled out sufficiently explicitly as to be able to take on the task of guiding feature selection. This turns out to be much more than merely a 'computational overhead' brought out by using the theory for a computational purpose: it in fact provides a natural lead on to consider discourse semantics and its formalization, because without this the grammar cannot be used.

The task of driving grammatical feature selections has been explored in computational implementations of SFGs in four main ways:

- by means of the *chooser and inquiry* semantics developed within the Penman text generation system (Mann 1985),
- by the *register-driven* semantics developed within the SLANG generation system (Patten 1988),
- by *probabilistic weightings* of feature selections within the COMMUNAL system (Fawcett & Tucker 1990),
- and by means of *constraint relations* as explored within several more experimental approaches to generation (Bateman *et al.* 1992; O'Donnell 1994).

Of these, the approach to 'inter-stratal' relationships pursued within the Penman system was the first extensive experiment and is, accordingly, probably the least systemically oriented, although it does bring certain benefits of its own to which we will return in a moment. Here again we have an example of how computational instantiation can drive linguistic theory development: the precise mechanisms by which strata are to interact have primarily been described abstractly; the computational instantiation allows us to explore just to what extent these proposals are adequate. Most traditionally in SFL, features at one linguistic stratum have been said to 'preselect' (directional metaphor) or 'meta-redound with' (non-directional metaphor) features at less abstract strata. This is the model explored by Patten: selections from a register network are aligned with selections from the lexicogrammar. This appears effective only for restricted cases; to obtain sufficient constraint in the lexicogrammar, the registerial features needed to be extended in delicacy to such an extent that it almost appeared to duplicate the information

required by the lexicogrammar. It turns out not to be straightforward to maintain the sense of lexicogrammar serving as a meaning potential within this model of inter-stratal interaction alone; this may have been due an inbuilt directionality in Patten's computational implementation. The approach in Fawcett and Tucker's model also attempts to embody certain traditional SFL concerns—in their case the treatment of a System network as a probabilistic system; here again there turn out to be complications that render the inter-stratal 'control' of a lexicogrammar difficult.

The chooser and inquiry framework, in contrast to these, was an explicit response to the computational requirement that the overall generation system be modular and re-usable across different contexts and across different computational systems, different knowledge representation languages, and different text planning components. This was achieved by associating a small 'choice expert', or *chooser,* with each grammatical system in the system network. A chooser breaks down the choice among the grammatical features of its associated system into one or more semantic questions, called *inquiries*. Inquiries take parameters that, typically, refer to aspects of the meaning, concepts, etc. that need to be expressed. It is the responsibility of these inquiries to obtain the information relevant for the grammatical decision. As far as the grammar and choosers are concerned, therefore, the inquiries represent semantically aware oracles which can be relied on to motivate grammatical alternations appropriately for the current communicative goals being pursued without themselves knowing anything about grammatical organisation. The chooser and inquiry approach has proved itself useful not only for modularizing the generation process, but also for providing a bridge between more informal, functional-descriptive approaches to linguistic phenomena of the kind often found in text linguistics and more formal approaches that are the goal of computational instantiation. This is important both for opening up a dialogue between computational and non-computational (functional) linguistics and for allowing sentence generation to proceed even in areas where our understanding of the mechanisms involved are still too incomplete to admit of detailed formalizations. In working out just how some particular semantic 'oracle' is to work out its answer, one is naturally led to consider all relevant linguistic research that can bear on the issue. The place where this has had clearest consequences has been in the provision of a detailed experiential semantics that takes on the task of answering all oracle questions arising within the experiential metafunction. This experiential semantics, originally called the Upper Model and recently developed considerably further by Halliday & Matthiessen (1999), is now at the centre of several explorations of linguistic semantics. Now that this exists, researchers can return to explore the precise nature of interstratal relationships more effectively.

The Penman system has served as the foundation for many subsequent investigations of systemic theory and its applications in computational contexts. In order to provide higher-level sources of control, the approach to discourse representation called Rhetorical Structure Theory (RST) was developed by Mann, Sandra Thompson and Matthiessen (Mann & Thompson 1988) and this later became an established mechanism for automatic text planning within NLG generally (Hovy 1993); a critical comparison of RST and Conjunctive Relations has been given by Martin (1992).

Several subsequent NLG systems also took the Penman system as their implementational starting point: i.e., the grammar of Nigel, or the code of the Penman system, or both were built on in order to provide additional functionalities.

For example, the original Penman system separated out grammar and lexicon as is common in natural language systems: the lexicon is then a separate list of words with

lexical, morphological and grammatical information attached. Marilyn Cross, in her 1992 dissertation from Macquarie, investigated to what extent a systemicized lexicogrammar would be feasible within the same framework (Cross 1992). Her system produced texts concerning the water cycle and, although successful as a prototype system, also demonstrated that the sheer descriptive work required by an integrated lexicogrammar in the style pioneered by Hasan (1987) is a severe computational challenge as well. The pursuance of lexicogrammar to the delicacy required for lexis is also a main orienting feature of the computational grammars of the COMMUNAL project (Tucker 1996, 1998). There remain many important theoretical issues to be resolved here, which the explicitness required of a computational account does not allow us to shirk. The combination of highly delicate lexical systems with inter-stratal control is a particularly difficult problem.

The Penman system was also substantially extended and modified to incorporate the results of discussions originally involving Halliday, Matthiessen, Bateman, Licheng Zeng and Keizo Nanri (Bateman *et al.* 1991) concerning the phenomenon of multilinguality. From the early 1990s on, multilinguality in linguistic description became an important issues in many approaches to NLG and many systems now attempt *multilingual natural language generation* (MLG). Here the functional orientation to language description promised a highly effective approach to constructing computational lexicogrammars and semantics for a broad range of languages. After some individual specifications for other languages expressed within the broad Nigel tradition using Penman, particularly German within the KOMET project in Darmstadt (Teich 1999) and German, French and Russian within the TechDoc project in Ulm (Rösner & Stede 1994), a general view of multilingual systemic networks and of more or less congruent perspectives induced by language varieties was added as a basic component of the definition format for system networks. The resulting system, KPML (Komet-Penman Multi-Lingual) (Bateman 1997), has continued to grow and is now one of the standard large-scale generation systems available for general use, including substantial computational grammars for Dutch, Chinese, Russian, Czech, Bulgarian, Spanish, as well as several smaller fragments. With large-scale resources, issues of maintenance and usability also become increasingly central. There have also been experiments and developments that have been pursued independently of the Penman system, although often drawing on its experience. Most significant here is O'Donnell's WAG system (O'Donnell 1996), which drew on advances in formalization to allow not only the analysis that we saw in the previous section but also generation similar to that performed by Penman. The WAG system was subsequently employed in a number of generation projects, such as the ILEX project at Edinburgh which relies on systemically-driven generation for the dynamic construction of webpages concerning museum exhibits, and supported the development of compatible computational grammars for further languages, most particularly Greek. There have also been moves towards extending generation to include multimodal presentations, supported by some exploratory systems as Zeng's MULTEX system (Matthiessen *et al.* 1995).

One further NLG system, that was developed in parallel and independently to the Penman system but which has also played a dominating role in many branches of NLG research and development, is Michael Elhadad's FUF (Functional Unification Formalism) (Elhadad & Robin 1996). FUF is an extension of Kay's FUG that we saw above specially tailored for generation. FUF is relevant to our discussion here because it also provides a very large generation grammar for English that is related to SFL,

and particularly to aspects of Fawcett's approach to transitivity. The development of this grammar has followed quite a different path, however, and integrates within it significant aspects of other linguistic theories, such as HPSG. The full impact of such combinations of grammatical approaches remains to be evaluated, although it can already be seen that the FUF treatment of English is very extensive.

In summary, we can see that both the research field of NLG and the techniques for information presentation that it provides are nowadays finding ever more possibilities of application. Established examples include the generation of weather reports from meteorological data and the generation of letters responding to customer queries. Potential applications now include for instance: the automatic production of technical documentation, of instructional texts, of patent claims, of computer-supported cooperative work, of patient health information and education, of medical reports and in natural language interfaces to databases and information systems. Systemic-functionally based generation has established a niche for itself within NLG, but its continued influence and development there will depend on several factors. There are many rival technologies and frameworks and it is no longer the case, as it was when the Penman system began, that these rivals are linguistically naive or computationally simple.

The main problems facing the adoption of systemically-based generation systems are similar to those facing NLG systems in general--there is a question of what additional functionality can we provide to counterbalance the quite substantial costs of development. Producing and maintaining a large-scale grammar of a language is an expensive and time-consuming activity. There is much discussion of whether it is not easier to have simpler, more *ad hoc* solutions to tasks--after all, if the user does not see a difference between a sophisticated and linguistically-motivated computational system and a few well-placed print statements of the form "print 'Your account has expired.'", there will be few willing to provide the funds for further linguistic research. NLG is at its best when the texts produced must reflect complex and changing relationships that require more of the language produced than a simple print statements can provide. Here, then, we may still found computational applications of systemic theory where much can be gained: particularly in the areas of register-controlled language generation, varying texts according to the tenor relationships involved (expert-novice, patient-doctor, customer-client, etc.). Each of these areas will require developments both in the computational modelling and in the state of the linguistic theory modelled.

Perhaps here we will find in the near future a similarly beneficial synergy as already found within the strata of lexicogrammar and semantics. Although, it must be said, we still have far too few systemically-trained linguists who could take up the computational challenge and without them, other solutions will be found.

For the development of SFL, we have seen that computational generation systems radically improve our abilities to test out large-scale grammar fragments to see if they are internally consistent and describe the linguistic structures that they were intended to. Much has been learnt about precisely what kinds of realizational constraint are necessary for constructing systemic grammars and their limitations. More detailed accounts of inter-stratal relationships and the interplay between information presentation in mutually supportive but distinct modalities are just beginning to come into play. There too we can expect a rich continued interaction between theory and computational application.

5 Dialogue Systems

A further context for interactions between SFL and natural language processing by computer is that of dialogue systems. Dialogue systems are computational components that engage a human interlocutor in dialogue in order to achieve some task. There are already some spoken-language systems in everyday use--for example, automatic train or bus timetable services--but these are generally characterised by rather weak interactional possibilities. Dialogue system research and development within computational linguistics seeks to construct systems whose interactional abilities approach that of human conversation far more closely, and this naturally provides important further potential for SFL, engaging with its accounts of prosody, discourse semantics and exchange structure in addition to the bias towards primarily lexicogrammatical accounts that we have seen so far.

In one of the earliest direct applications of SFL in this area, Telecom Australia and a group at the University of Sydney started a research project in 1991 to develop a telephonic dialogue system: a system which would allow human callers to speak to a computer 'operator' to obtain information. For the dialogue component, the dynamic model of exchange proposed in O'Donnell (1990) was adopted, and an implementation put together by Telecom. This model adapted the Berry model of exchange, along with elements of Martin's model. It consists of two components: a network of exchange states (the 'context' in the current exchange, e.g., *proposition-completed-but-not-supported*), and a set of behavioural (move) options for each participant. The behavioural options are conditioned by the exchange states, and the enactment of a move causes a change in the exchange states.

A second model (cf. O'Donnell & Sefton 1995) described the potential generic tasks that need to be performed in an information-seeking dialogue (e.g., *Greet, State-Information-needed*, etc.), and a network similar to the exchange state network conditioned these options.

The project also produced reports detailing how the systemic model of exchange could be mapped down into grammar, and then to speech. However, none of this was implemented. Telecom used a phrase-structure grammar for syntactic analysis and generation. Due to the 1992 recession, the project was terminated before a full prototype could be implemented

Another project, this time involving dialogue capabilities in German, was later undertaken in cooperation between the KOMET generation group in Darmstadt and the Technical University of Budapest. The goal of this project was to explore situationally and interactionally appropriate intonational control for German utterance generation (Teich *et al.* 1997). Again, a dialogue model based on state-transitions was employed so that the generation component always knew both where it was in an exchange and what particular questions were at issue. The natural language component focused on the generation task while another group from Darmstadt developed a graphically based interface for an information retrieval system. The input to the dialogue component was then situated mouse-clicks and typed text, while the output was natural spoken German with intonational contours based on Pheby's (1969) adaption for German of Halliday's account of English prosody. Adding intonational resources to a computational descriptions emphasises some difficult theoretical issues. For example, just how the potential non-congruence of information unit and clause is to be handled technically remains an unsolved question. The fine alignment of the prosodic contour with the syntactic and lexical material is also

problematic. Speech synthesis presents a very useful challenge for developing such models further because the results of a faulty model can generally be heard all too clearly. More recently, there has also been work that attempts to reconcile, or at least relate, the approach to SFL approach to intonation with the tone-based accounts that are prominent in non-systemic computational work (Teich *et al.* 2000); such work is very important for extending the breadth of interactions in the field.

One further project is currently developing a dialogue system for Japanese interaction (Kobayashi *et al.* 2002; Ito *et al.* 2004). This system, based at the Japanese Brain Institute RIKEN under Michio Sugeno, draws extensively on the Systemic interaction models being developed at Macquarie University. The essential mechanism is that features selected from network descriptions of context and the interactional situation preselect lexicogrammatical features for guiding the generation of linguistic utterances. Similarly, particular syntactic fragments that are recognised in a user's utterances trigger the selection of interactional and context features. There remain many difficult issues here that are beyond the scope of the present chapter; in particular, it is unclear at this time both just how well the dynamic development of the interaction will be modelled and how effective the analysis will prove to be. Dialogue systems are a hard test for any linguistic theory: the respective problems of analysis and generation are compounded and several further areas are required. Moreover, the resulting systems need to function with real, and often surprisingly varied, user input and in something approaching 'real time'. And, as we shall see in the section following, there are some reasons to doubt whether Systemic accounts in their present form can provide all that is necessary here. This is a clear indication that further, closer exchanges between Systemic theory and computation may be highly beneficial.

6 Representation and formalisms

Whereas the previous sections of this chapter have described the computational contexts of use that SFL has found, there is a further significant area of interaction between SFL and computation that cannot be so immediately tied to particular tasks. Earlier approaches to artificial intelligence and natural language processing were often content if they could construct a computer program that appeared to perform as required; this also applies to most of the computational instantiations of SFL that we have seen so far. However, as computational linguistics has matured, it has became clear that one should also consider the abstract computational properties both of the representations that are employed for linguistic information and of the tasks undertaken.

This involves the construction of *formalisms* that are used to represent any linguistic information required. A formalism has a specified formal semantics that states precisely both how expressions in the formalism are to be interpreted and how consequences can be drawn on the basis of those expressions. Such consequences are themselves generally stated in terms of computable inferences, i.e., algorithms or computer programs, that take expressions in the formalism as input and produce further expressions in the formalism as output. This degree of precision brings many benefits; for example, it becomes possible to demonstrate that a particular task will always be achievable (or not!) regardless of the particular input faced. This is then a significant step beyond writing a program that appears to work for the inputs that one has tried so far--one can be sure of the behaviour of the program for any input.

This is also important linguistically; applying appropriate formalisations of, for example, Systemic networks means that we can demonstrate that a network is coherent and that the results of combining particular sets of realization statements entailed by a selection expression (path through the network) is guaranteed to result in a syntagmatic expression. The effort involved in constructing a full formalization of a linguistic account can often reveal gaps or places in the conceptualization of the linguistic representation that were not previously clear. One of the earliest examples of the importance of this in linguistics was the demonstration of Peters and Ritchie (1993) that, *regardless of how it was expressed*, adopting a transformational grammar of the original Chomskian type would commit the linguist to accepting an unacceptably high degree of computational complexity. The very nature of the formalism and theory had inherent properties that determine how it can perform for particular tasks: indeed, certain consequences of linguistic statement made within the theory were shown to require an arbitrarily long time to compute.

It is quite important to understand what this means when considering computational models. The 'arbitrarily long time' does not refer to the kind of performance issues that might be addressed by getting a faster computer; high computational complexity refers to performance that deteriorates at a rate that radically outruns *any* increase in processing time. Thus, if your formalism and the consequences that it is intended to support run foul of such higher degrees of computational complexity, then this is a serious matter: the theory is then 'too powerful' to stand as a model of the behaviour modelled since there is no possibility of defining algorithms or building processors that can carry out that intended theory. The issues are rather more subtle than can be presented here; indications of computational complexity can in fact be employed constructively--for example, by dividing problems up into subcomponents so that the real sources of complexity can be isolated and perhaps resolved in other ways. But the main point of the discussion is simple: relating linguistic representations to computational formalisms can reveal significant information about the inherent 'computability' (and this includes 'computability' by processors such as our brains) of those linguistic representations and theories constructed in their terms.

For SFL, too, this is a significant issue. It turns out that a straightforward rendering of Hallidayan Systemic grammar of the kind expressed in the Nigel grammar (see above) suffers exactly from these higher levels of computational complexity. This cannot be changed by rewriting the grammar in clever ways; the *form of the grammar itself* already invokes a high degree of computational complexity. This is the underlying reason why all of the attempts to produce automatic analysis components for a Systemic grammar, as we described above, have needed to fall back on various kinds of 'simplifications'. It seems that a Systemic grammar written solely within the representational possibilities provided by the Nigel grammar cannot support fully general automatic analysis.

This inherent complexity of Systemic representations has been known for some time and a number of researchers have attempted to define restricted versions of the 'Systemic formalism' possessing more attractive computational properties. This involves first understanding in detail what the essential descriptive properties of a Systemic grammar are and investigating whether aspects of this can be left out, or cut down, so that what is left is still useful while not being computationally as powerful. Success here would have had immediate practical applications. For example, if a restricted version of a Systemic formalism were found that still allowed Systemic

linguists to write the grammars they want to, then such grammars would be directly applicable to automatic analysis. Unfortunately, this search does not look hopeful.

Most of the early explicit attempts to formalize, i.e., to provide a formalism for, Systemic grammars focused on their paradigmatic organization. It was correctly seen as essential that a proper formal representation of a Systemic grammar should allow delicacy-related inferences to be drawn: that is, it should be possible from any feature in the network to infer automatically just what other features are necessarily entailed, necessarily rejected, and which were still potentially compatible. For a simple taxonomic classification network, this is not a difficult task; but for a Systemic network, which allows complex entry conditions and simultaneous systems, it is a major source of computational complexity. This was described, for example, by Patten & Ritchie (1987) using a logic representation and by Mellish using Kay's notion of unification (see above) (Mellish 1988). Subsequently, Brew (1990, 1991) attempted to reduce the computational complexity by restricting the kinds of networks that can be written. This succeeded in reducing complexity but at the same time appeared to reject precisely the kind of network connectivity that one finds most often in detailed Systemic descriptions.

We have already noted in Section 3 above on analysis how Kay's Functional Unification Grammar from 1985 can be seen as an approach to formalising a Systemic syntagmatic structure. As this formalism became more widely used in computational linguistics and was applied to broader bodies of linguistic description, it became clear that the use of the FUG-expressions could be improved by organising them into inheritance hierarchies similar in form to Systemic networks. The resulting family of formalisms were called Typed Feature Structures because each FUG-expression was allocated to a *type* and these types were organised into a special form of inheritance hierarchy, corresponding to a mathematical structure called a *type lattice*. In many respects this provides the most natural formalisation of a Hallidayan-style Systemic grammar and several variants along these lines have been proposed; the earliest, which we described above, also served as experiments in parsing and analysis using Systemic grammar. Within these approaches, the 'types' naturally correspond to grammatical features in a Systemic network and the functional unification structures (see Figure 2) correspond to the syntagmatic information expressed in realization statements.

Such representations are completely independent of their use and so also capture well the 'non-directional' aspect of a Hallidayan Systemic grammar. The formalism can support inferences from less abstract strata to higher, i.e., from lexicogrammar to semantics for interpretations, and from higher strata to lower, i.e., from semantic to lexicogrammar for generation. But this power has a high cost--namely, the computational complexity alluded to above. In the mid-1990s, in the last large-scale work to be carried out in this area, Renate Henschel examined the major tools for using typed feature structures that were available and explored how each of them fared when confronted with a typed feature representation of the full Nigel grammar of English. The results, reported by Henschel (1994, 1997), showed that only one system at that time could even load the entire grammar and that inferences drawn on the basis of that grammar were prohibitively slow. There are now very much more efficient and robust tools for using large-scale typed feature structures available. Although these have been developed primarily for the rather differently organised grammars of HPSG (Copestake 2002), it is perhaps time again to explore to what extent these might also support Systemic grammars.

7 Systemic Coding Tools

The previous sections have explored computer systems which apply SFL to some task, such as text generation or analysis. This section and the next will look at tools which help systemicists do their work. This section will describe tools which enable the linguist to hand-analyse (or 'code') a text.

7.1 Systemic Coder (O'Donnell)

During the EDA project (1992, see section 3.8 above), the researchers required a parsed corpus of computer manuals in order to judge which parts of the grammar were most utilised in the manuals. To this end, a tool was developed which allowed a linguist to step through a network, choosing a feature at each step, that for the current text segment.

Since then, the tool has been continually under development, oriented more towards helping linguists perform statistical studies of text. Watching users utilise the tool, the author has seen where they have trouble, and has refined the tool. Currently, users report it is easy to use yet powerful.

The current version includes a graphically displayed system network representing the coding categories, a screen displaying the text, which is also where the user indicates the segment boundaries (e.g., clause, group, etc.). Another interface allows limited corpus search: retrieving all segments which contain a given feature, or logical combination of features. A fourth interface is used for statistical analyses of the corpus, such as contrasting two subsections of the corpus, or deriving descriptive statistics.

The coder automates some processes for the user. Firstly, segmentation of the text into sentences or paragraphs is automatic (the user needs to add in segment boundaries at other levels). An experimental version even does the coding for the user: given a corpus of already-coded segments, the programme uses machine learning techniques to predict which coding choices are appropriate for any new segment. Reliabity varies between 50% and 80%, which is why the auto-coder is still experimental.

The Coder has two main limitations. Firstly, only one level of analysis can be performed within a single file, e.g., one cannot code segments at clause level, and also code the groups within the clauses. For that, a copy of the text needs to be taken, and a separate coding file made. This makes it difficult to explore relations between ranks, e.g., what types of groups function as Subject in certain clause types. Secondly, the tool does not properly handle embedded segments, e.g., to code both clauses in "He had, I think, two options".

He is currently working on a new tool which fixes both these problems. Also, the Coder was originally designed to deal with single text files, but more and more, it has been necessary to deal with a corpus of text files, each coded according to the same scheme. To this end, the ability has been added to the Coder to load a set of coding files at the same time, and do corpus search and statistics as if they were one document. However, this is not yet an optimal solution. In the new tool, the basic assumption is that the user is working on a document set, a 'project', making the performance of large studies more intuitive.

7.2 Functional Grammar Processor (Webster)

In 1994, Jonathan Webster announced his Functional Grammar Processor (FGP), a tool for functionally analysing clauses in terms of either Transitivity, Mood or Theme structures, It was integrated into a word-processing package. The user opened up a text in the package.

> "The user then highlights the clause to be analyzed and selects from the FPG pop-up menu to do either Theme-Rheme, Mood-Residue or Transitivity analysis ... The FG Processor ... next appears on the screen. The user enters his/her analysis into the appropriate fields." (Webster 1994: 183).

An online help system was available to help an unsure user: for instance, if not sure which sub-type of relational to code the verb, the help menu glossed the options.

FGP only allowed the user to analyse within the defined parameters of the system, while the Systemic Coder allowed the user to create their own schemes. Also, FGP is limited to analysing clauses. However, it does allow embedded clauses to be analysed quite easily.

7.3 SysFan (Wu, Matthiessen)

SysFan was developed by Canzhong Wu, at Macquarie University, under the direction of Christian Matthiessen, in the late 1990s. It is closer to Webster's tool than the Systemic Coder, in that it codes with pre-loaded model of language, and the usual user just analyses text according to that model (experts can in fact change the model though).

Like the Coder, the tool starts off with a segmentation interface, where the user segments their text into sentences. The user can then open each sentence in another interface, where they further divide the sentence into component clauses, and can assign clause complex structure to these clauses (both structurally, as in relations of α, β, 1, 2, 3, etc., and also paradigmatically in terms of the features, e.g., *hypotactic:projection:idea*. The tool includes a nice innovation in that it allows the user to assign features simply by clicking on the leaves of a graphically drawn system network.

The tool then allows each clause to be analysed into Transitivity, Mood and Theme structures (both structurally and paradigmatically). Again, the use can select the features directly off a system network. SysFan allows the user to view various profiles over the text, such as what types of processes make up the text, and which participant types they use.

The tool appears to be excellent as a means of analysing text according to the loaded model. The main constraint is that it is only available for Macintoshes.

7.4 'Systemics' (O'Halloran & Judd)

Another recently developed coding tool is 'Systemics' designed and implemented by Kay O'Halloran and Kevin Judd (O'Halloran 2003). This system is unique among Systemic computational tools in that it attempts to provide a graphically based annotation interface for several quite distinct kinds of linguistic information, each traditionally with its own style of graphical presentation. Codings are possible for grammatical structures, interclausal logical structure in terms of elaboration, enhancement, extension and projection, dialogic structure in terms of Berry's

exchange structure, Martin's conjunctive relations, and lexical cohesive chains as developed by Hasan and Martin. Each is shown in the appropriate graphical form, and this is also directly editable.

In addition to this, all linguistic resources are maintained as Systemic networks and their content is also directly editable; thus, while the system comes with a substantial grammar for English pre-loaded, it is possible to edit this further for one's own purposes. The pre-loaded grammar is also unique in that it comes with a description for each of the choices and grammatical functions that are included, plus references to further reading.

Somewhat non-traditional is the display metaphor selected for the linguistic resources; these resemble the file explorers of modern operating systems such as Windows, where one can open and close folders, following these down to greater or lesser depths. There is no graphical network display. All the linguistic information is expressed in this form, regardless of whether it is paradigmatic or syntagmatic. Whether this is simpler to follow and use than the traditional graphical displays with its separation of paradigmatic (network) and syntagmatic (realization statements) information is still untested.

Finally, 'Systemics' is the first tool for Systemic linguists that it is not free. Funding to develop the software was made available from the University of Singapore Press, under the condition that the software would only be available under license. The current single-user licence fee is around S$100. Although potentially a very good tool for students, it is unclear whether the market is already sufficiently strong for this overhead.

7.5 KPML-based tools (Bateman)

Whereas the other tools described in this section are essentially task-oriented, that is they try and provide tools for particular tasks that it is known are useful for the working Systemic linguist, the tools developed on the basis of the KPML generation system (see above), are more exploratory. Taking particularly the suggestions for new styles of linguistic grammars made by Matthiessen & Nesbitt (1996), the aim is to support the linguist by providing differing views of both linguistic resources and instances, i.e., texts, combining and moving between perspectives. Many of these perspectives requires that the tool has a rather complete 'understanding' of the linguistic resources that it contains. For example, it is essential to know the consequences of the system network, that some features can occur together and entail others, whereas other features might be mutually exclusive--this prevents potentially inaccurate codings. Whereas such requirements are quite strong when seen from the perspective of the tool builder designing a tool afresh, they are already provided by a full text generation system such as KPML or Penman. If the text generation system did *not* have this information available to it, it would not be able to follow the instructions provided by a specification of the grammatical potential to produce instantiated syntagmatic structures. The KPML tools, including graphers, grammar documentator, example manager, and coder, are all based on the simple strategy of making information visible that is already maintained internally within the generation system. This makes it quite simple to provide a host of sophisticated views, such as for example, highlighting all examples from a set of sentences where some specified pair of grammatical constituents (functions) are conflated in the context of some particular combination of paradigmatic features, profiling particular selections of

features or participant roles, or overviewing the grammatical functions and the realization statements operating over them for some selected functional region of a grammar network, and many more. The interesting questions then revolve more around finding out just what views are useful for a linguist or student or casual browser and which are not.

The more that linguists are exposed to these kinds of tools, the more we can expect usefully tailored functionalities to be provided. But the first step here will be a willingness and curiosity on the part of linguists to see what the computational tools that are being explored can offer.

8 Presentation

As with other applications, tools for drawing and editing system networks have been evolving over time. Here also, we see a movement from tools which required computational skill, towards higher usability by linguists.

During the mid-1980s, Michael Cummings in Toronto developed a tool called SYSPRO (Cummings 1987), which allowed the user to enter system networks using a command line interface, and then display the network as a whole, printed using fixed-width characters, for texts and brackets.

At ISI, home of the Penman project, Gabriel Robins programmed a grapher for drawing inheritance networks, although in a very non-systemic form. This grapher was used within Penman.

Based on his algorithms, Mick O'Donnell developed a grapher which drew more Systemic diagrams (1990), which was used for displaying grammars within his WAG system. However, the network needed to be edited in a text editor.

Kumano *et al.* (1994) also developed a grapher for Systemic grammar, and incorporated into it graphical means for editing the system network. While the O'Donnell grapher automatically lays out the graph, this grapher requires the user to click on the location on the canvas where they want the system. The advantage of this approach is that the user has more control of the graph, but with large grammars, a change in one system may necessitate moving many systems.

In 1999, O'Donnell extended his grapher to allow graphical editing of the network as well. This is now incorporated into various tools, such as his Coder software and used for editing grammars for his parsers and generator. A Java version of the tool is being integrated into the KPML Text Generation system.

As part of the SysAm Toolbench, Canzhong Wu (Macquarie University) also developed a grapher for system networks (late 1990s), So far, the network is not editable directly, rather the user edits the systems within a text file. However, his grapher is utilised as a means of selecting selection expressions within his SysFan coding system (see above).

9 Conclusions

The long interaction between SFL and computation has enriched both sides. For SFL there are some particular lessons to be learnt and challenges to be met. It is interesting, for example, that SFG, when used for text generation, favours rich, functionally complex, descriptions. Yet when we apply SFG to parsing, the pressure

is to reduce the richness, move towards a grammar which, at its first bite, deals most directly with form. However, this is not to say that we need to imitate the formal approach of the post-Chomskian models: if language has evolved in the expression of meaning, then we should expect the forms of language to be organised in meaningful ways. However, language has also evolved to be learnt by infants, and as such, the patterns of language cannot be so complex that this is an impossible task. But our current SFGs say very little about the patterns of language that are being learned: their accounts of syntagmatic configurations and instantiated resources is still very exploratory. The truth is that we need deeper accounts of both of these two factors. If we are to use the same SFG for both the generation and analysis of text, we will need to enrich it to contain both the functional complexity that we are familiar with and the relative formal simplicity that makes parsing, by people and machines, the robust and reliable facility that it is.

References

Bateman, John, Christian M. I. M. Matthiessen, Keizo Nanri & Licheng Zeng (1991). "The re-use of linguistic resources across languages in multilingual generation components." Proceedings of the 1991 International Joint Conference on Artificial Intelligence, Sydney, Australia. Vol. II. Morgan Kaufmann Publishers, 966-971.

Bateman, John, Martin Emele & Stefan Momma (1992). "The Nondirectional Representation of Systemic Functional Grammars and Semantics as Typed Feature Structures." Proceedings of COLING-92. Nantes, France, Volume III, 916-920.

Bateman, John, Robert Kasper, Johanna Moore & Richard Whitney (1990). A General Organisation of Knowledge for Natural Language Processing: the Penman Upper Model. USC/Information Sciences Institute Technical Report.

Bateman, John, Robert Kasper, Jörg Schütz & Erich Steiner (1989). "A new view on the translation process." Proceedings of the European Chapter of the Association for Computational Linguistics. Manchester, England. April, 1989. Association for Computational Linguistics.

Bateman, John (1997). "Enabling technology for multilingual natural language generation: the KPML development environment." Journal of Natural Language Engineering, 3(1): 15-55.

Brew, Christopher (1990). "Partial descriptions and systemic grammar." 13th. International Conference on Computational Linguistics (COLING-90). Helsinki, Finland, 36-41.

Brew, Christopher (1991). "Systemic Classification and its Efficiency." Computational Linguistics, 17(4) (Dec.): 375-408.

Charniak, E. (1996). ``Tree-bank grammars'' Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96). MIT Press, Cambridge, MA, 1031-1036.

Copestake, Anne (2002). Implementing Typed Feature Structure Grammars. CSLI Publications. Stanford, CA.

Cross, Marilyn (1992). "Choice in Lexis: Computer Generation of Lexis as Most Delicate Grammar." Language Sciences, 14(4): 579-607.

Cummings, Michael & Al Regina (1985). “A PROLOG parser-generator for Systemic analysis of Old English Nominal Groups”, in James Benson & William Greaves (eds.) Systemic Perspectives on Discourse: Selected Theoretical Papers from the 9th International Systemic Workshop. Norwood, N.J.: Ablex.

Cummings, Michael (1987). “Syspro: a computerized method for writing system networks and deriving selection expressions.” In Ross Steele & Terry Threadgold (eds.), Language topics: essays in Honour of Michael Halliday. Amsterdam: Benjamins.

Davey, A. (1978). Discourse Production: A Computer Model of Some Aspects of a Speaker. Edinburgh University Press. Edinburgh.

Elhadad, Michael & J. Robin (1996). “A reusable comprehensive syntactic realization component.” In: Demonstrations and Posters of the 1996 International Workshop on Natural Language Generation (INLG '96), 1-4. Herstmonceux, England.

Fawcett, Robin & Gordon Tucker (1990). “Demonstration of GENESYS: a very large, semantically based systemic functional grammar.” 13th. International Conference on Computational Linguistics (COLING-90), Vol. I. Helsinki, Finland, 47-49.

Fawcett, Robin (1973). “Generating a sentence in systemic functional grammar.” In M.A.K. Halliday & James R. Martin (eds.) Readings in systemic linguistics. Batsford, London.

Halliday, Michael A. K. & Christian M. I. M. Matthiessen (1999). Construing experience through meaning: a language-based approach to cognition. Cassell. London.

Hasan, Ruqaiya (1987). “The grammarian’s dream: lexis as most delicate grammar.” In M. Halliday & R. Fawcett (eds.) New developments in systemic linguistics: theory and description. Pinter, London.

Henrici, Alick (1965 “Some notes on the Systemic generation of a paradigm of the English clause.” Working Paper for the O.S.T.I. Programme in the Linguistic Properties of Scientific English. Reprinted in M.A.K. Halliday & James R. Martin (eds.) 1981 Readings in Systemic Linguistics, London: Batsford.

Henschel, Renate (1994). “Declarative Representation and Processing of Systemic Grammars.” In Carlos Martin-Vide (ed.) Current Issues in Mathematical Linguistics, 363-371. Elsevier Science Publisher B.V., Amsterdam.

Henschel, Renate (1997). “Compiling Systemic Grammars into Feature Logic Systems.” In Suresh Manandhar, Werner Nutt & Gabriel Pereira Lopez (eds.) CLNLP/NLULP Proceedings.

Hovy, Eduard H. (1993). “Automated discourse generation using discourse relations.” Artificial Intelligence, 63(1-2): 341-385.

Hudson, R.A. (1971). English complex sentences; an introduction to systemic grammar. Amsterdam, North-Holland Pub. Co.

Hutchins, John (1986). Machine translation: past, present, future (Ellis Horwood Series in Computers and their Applications). Chichester, Ellis Horwood.

Ito, Noriko, Toru Sugimoto & Michio Sugeno. (2004). “A Systemic-Functional Approach to Japanese Text Understanding.” In Alexander F. Gelbukh (ed.)

Computational Linguistics and Intelligent Text Processing, 5th International Conference (CICLing 2004), 26-37. Springer.

Kasper, Robert (1988). "An Experimental Parser for Systemic Grammars." Proceedings of the 12th Int. Conf. on Computational Linguistics, Budapest, Association for Computational Linguistics.

Kay, Martin (1979). "Functional Grammar." Proceedings of the Fourth Annual Meeting of the Berkeley Linguistics Society.

Kay, Martin (1985). "Parsing in Functional Unification Grammar." In D. Dowty, L. Karttunen, & A. Zwicky (eds.), Natural Language Parsing, Cambridge University Press, Cambridge, England.

Kobayashi, Ichiro, Moon-Soo Chang & Michio Sugeno (2002). "A study on meaning processing of dialogue with an example of development of travel consultation system." Information Science, 144(1): 45-74.

Kress, Gunther, Ruqaiya Hasan & James R. Martin (1992). "Interview--M.A.K. Halliday May 1986." Social Semiotics 2.1 (May 1992): 176-195.

Mann, William C. & James A. Moore (1980). Computer as Author -- Results and prospects. Technical report. Information Science Institute. USC/Information Sciences Institute, Marina del Rey, CA.

Mann, William C. & Sandra A. Thompson (1988). "Rhetorical Structure Theory: Toward a Functional Theory of Text Organization." Text, 8(3), pp. 243-281.

Mann, William C. (1985). "An introduction to the Nigel text generation grammar." In James Benson & William Greaves (eds.) Systemic Perspectives on Discourse: Selected Theoretical Papers from the 9th. International Systemic Workshop, 84-95. Ablex Pub. Corp., Norwood, N.J.

Martin, James R. (1992). English text: systems and structure. Benjamins. Amsterdam.

Matthiessen, Christian & William C. Mann (1985). "Demonstration of the Nigel Text Generation Computer Program." In J. Benson & W. Greaves (eds.) Systemic Functional Approaches to Discourse. Ablex, Norwood.

Matthiessen, Christian M. I. M. (1985). "The systemic framework in text generation: Nigel." In James Benson & William Greaves (eds.) Systemic Perspectives on Discourse, Volume 1, 96-118 Ablex, Norwood, New Jersey.

Matthiessen, Christian M. I. M., I. Kobayashi, L. Zeng & M. Cross (1995). "Generating multimodal presentations: resources and processes." Proceedings of the Australian Conference on Artificial Intelligence. Canberra.

Matthiessen, Christian M.I.M. & Christopher Nesbitt (1996). "On the idea of theory-neutral descriptions." In Ruqaiya Hasan, Carmel Cloran & David Butt (ed.) Functional descriptions -- theory in practice, 39-85. Benjamins, Amsterdam.

Matthiessen, Christian, Michael O'Donnell & Licheng Zeng (1991). "Discourse Analysis and the Need for Functionally Complex Grammars in Parsing." Proceedings of the Second Japan-Australia Joint Symposium on Natural Language Processing, October 2-5, 1991, Kyushu Institute of Technology, Iizuka City, Japan.

McCord, Michael (1975). "On the form of a systemic grammar." Journal of Linguistics, 11: 195-212.

McCord, Michael (1977). “Procedural systemic grammars.” International Journal of Man-Machine Studies, 9: 255-286.

McCord, Michael (1980). “Slot Grammars.” American Journal of Computational Linguistics, Volume 6, Number 1, January-March, 31-43.

Mellish, C. (1988). “Implementing Systemic Classification by Unification.” Computational Linguistics, 14(1): 40-51.

O’Donnell, Michael & Peter Sefton (1995). “Modelling Telephonic Interaction: A Dynamic Approach.” Interface. Journal of Applied Linguistics 10.1: 63-78.

O’Donnell, Michael (1990). “A Dynamic Model of Exchange.” Word, vol. 41, no. 3, Dec. 1990.

O’Donnell, Michael (1993). “Reducing Complexity in a Systemic Parser”, in Proceedings of the Third International Workshop on Parsing Technologies, Tilburg, the Netherlands, August 10-13, 1993.

O’Donnell, Michael (1994). Sentence analysis and generation: a systemic perspective. PhD thesis. Department of Linguistics, University of Sydney, Australia.

O’Donnell, Michael (1996). "Input Specification in the WAG Sentence Generation System.” Proceedings of the 8th International Workshop on Natural Language Generation, Herstmonceux Castle, UK, 13-15 June 1996.

O’Donoghue, Tim F. (1991a). “The Vertical Strip Parser: A lazy approach to parsing”, Research Report 91.15, School of Computer Studies, University of Leeds, Leeds, UK.

O’Donoghue, Tim F. (1991b). “A Semantic Interpreter for Systemic Grammars”, in Proceedings of the ACL Workshop on Reversible Grammars, University of California at Berkeley, June 1991.

O'Halloran, Kay (2003). “Systemics 1.0: Software for Research and Teaching Systemic Functional Linguistics.” RELC Journal, 34.2: 157-178.

Patten, Terry & Graeme Ritchie (1987). “A formal model of systemic grammar.” In Gerard Kempen (ed.) Natural Language Generation. Martinus Nijhof, Dordrecht.

Patten, Terry (1988). Systemic Text Generation as Problem Solving. Cambridge University Press. Cambridge, England.

Peters, Stanley & Richard Ritchie (1973). “On the generative power of transformational grammars.” Information Sciences, 6: 49-83.

Pheby, John (1969). Intonation und Grammatik im Deutschen. Akademie-Verlag. Berlin.

Rösner, Dietmar & Stede, Manfred (1994. „Generating multilingual documents from a knowledge base: the TECHDOC project.” Proceedings of the 15th. International Conference on Computational Linguistics (COLING 94), 339-346. Kyoto, Japan.

Rowles C., M. de Beler, M. O’Donnell & P. Sefton (1993). “The Use of Context in the Understanding of Spoken English.” Proceedings of the 6th Australian Joint Conference on Artificial Intelligence, Melbourne, November, 1993.

Souter, Clive (1996). A Corpus Trained Parser for Systemic Syntax. Ph.D. Thesis. School of Computer Studies, University of Leeds.

Steiner, Erich, Ursula Eckert, Birgit Weck & Jutta Winter (1988). "The Development of the EUROTRA-D System of Semantic Relations." In Erich Steiner, Paul Schmidt & Cornelia Zelinksy-Wibbelt (eds.) From Syntax to Semantics: insights from Machine Translation. Frances Pinter, London.

Teich, Elke, C.I. Watson & C. Pereira (2000). "Matching a tone-based and a tune-based approach to English intonation for concept-to-speech generation." Proceedings of the 18th. International Conference on Computational Linguistics (COLING 2000). Saarbrucken, Germany.

Teich, Elke, Eli Hagen, Brigitte Grote & John A. Bateman (1997). "From communicative context to speech: integrating dialogue processing, speech production and natural language generation." Speech Communication, 21(1-2): 73-99.

Teich, Elke (1999). Systemic functional grammar in Natural Language Generation: linguistic description and computational representation. Cassell. London.

Thanning Vendelø1, Morten (2002). "An Interview with Terry A. Winograd." http://www.inf.cbs.dk/departments/inf/working-papers/papers_2002/2002-7.pdf

Tucker, Gordon (1996). "So grammarians haven't the faintest idea: reconciling lexis-oriented and grammar-oriented approaches to language." In Ruqaiya Hasan, Carmel Cloran & David Butt (eds.) Functional descriptions -- theory in practice, 145-179. Benjamins: Amsterdam.

Tucker, Gordon (1998). The Lexicogrammar of Adjectives: a systemic functional approach to lexis. Cassell. London and New York.

Weerasinghe, A. Ruvan & Robin Fawcett (1993). "Probabilistic Incremental Parsing in Systemic Functional Grammar." Proceedings of the Third International Workshop on Parsing Technologies, Tilburg, the Netherlands, August 10-13, 1993.

Wilcock, Graham (1993). Interactive Japanese-European text generation - an approach to multilingual export translation based on Systemic Functional Grammar. M.Sc. thesis. University of Manchester.

Wilks, Yorick (1995). "Arthur Frederick Parker-Rhodes: a memoir." Paper given as a Parker-Rhodes Memorial Lecture at the annual meeting of the Alternative Natural Philosophy Association in Cambridge, September 1995. (http://www.dcs.shef.ac.uk/~yorick/papers/cs-95-22.ps).

Winograd, Terry (1972. Understanding Natural Language, Academic Press.

[i] Previously, specific-purpose 'computers' were constructed for various tasks, such as vote counting. Only during the 1940s were general-purpose computers created, capable of storing a program loaded from an external source.

[ii] The EDA project also included Christian Matthiessen, Guenter Plum, Chris Nesbitt, Licheng Zeng and Arlene Harvey. The project produced very interesting results which were sadly under-reported at the time. Licheng Zeng produced a discourse-level analyser, which critiqued the multi-sentence structure of the text.