Probabilistic Incremental Parsing in Systemic Functional Grammar

A. Ruvan Weerasinghe*
Robin P. Fawcett
Computational Linguistics Unit, University of Wales College of Cardiff, UK.

July 5, 1993

Abstract

In this paper we suggest that a key feature to look for in a successful parser is its ability to lend itself naturally to semantic interpretation. We therefore argue in favour of a parser based on a semantically oriented model of grammar, demonstrating some of the benefits that such a model offers to the parsing process. In particular we adopt a systemic functional syntax as the basis for implementing a chart based probabilistic incremental parser for a non-trivial subset of English.

1 Introduction

The majority of the research in the field of Natural Language Understanding (NLU) is based on models of grammar which make a clear distinction between the levels of syntax and semantics. Such models tend to be strongly influenced by formal linguistics in the general Chomskyan paradigm, and/or by mathematical formal language theory, both of which make them conducive to computer implementation. Essentially, these models constitute an attempt to 'stretch' techniques that have been applied successfully to parsing artificial (and so unambiguous) languages, in order to apply them to natural language (NL).

In recent years, however, models of language that are derived from the text-descriptive tradition in linguistics have emerged as potentially relevant to NLU. These models emphasize the *semantic* and *functional* richness of language rather than its more formal and syntactic properties. Such models may challenge widely held assumptions, e.g. that the key notion in modelling syntax is grammaticality, and that this is to be modelled using some version of the concept of a phrase structure grammar (PSG)[1].Since such grammars emerge from use in analysing texts, they have something in common with the sort of grammars that tend to be used in corpus linguistics. To date the strongest influence of these grammars has been in Natural Language Generation (NLG) (Fawcett et al., 1993; Matthiessen, 1991).

The semantic orientation of functional grammars, however, is to some extent in conflict with the better understood techniques for parsing syntax. The research described in

*On leave from the Department of Statistics and Computer Science, University of Colombo, Sri Lanka.

[1]It is evident, however, that researchers working in the formal linguistics paradigm have in recent years increasingly realized the importance of the functional aspects of language, e.g. in augmenting their models with syntactico-semantic features.

this paper presents a probabilistic approach to parsing that yields a *rich syntax*, using a systemic functional grammar (SFG).In doing so, however, it also shows how some of the techniques used in traditional syntax parsing can be adapted to serve as useful tools for the problem. It will be shown that our parser is able to produce richly annotated ***flat*** parse trees that are particularly well-suited to higher level processing.

The main contributions to the formal specification of SFG, as they affect NLU, have been by Patten and Ritchie (1986), Mellish (1988), Patten (1988), Kasper (1987) and Brew (1991). These have mainly been concerned with the reverse traversal of system networks in order to get at the features from the items (words). They all conclude that systemic classification is NP-hard, but seek to isolate tractable sub-networks in order to be able to optimise reversal. It is thus apparent that a direct reverse traversal of the networks may not be the best approach to parsing in SFG.

Work of a more implementational nature is reported in Kasper (1988), Atwell et al. (1988) and Matthiessen (1991). The common approach to parsing systemic grammar in these has been to employ a 'cover grammar' for preprocessing the syntactic structure of the input string (instead of attempting to directly reverse the networks and realization rules), and then, as a second stage, to do the semantic interpretation by accessing the features contained in the system networks. O'Donoghue (1991b) suggests one possible way to avoiding this, namely by the use of a 'vertical strips parser'. This extracts the 'syntax rules' that are implicit in the grammar through analysing a corpus of text randomly generated by the generator (GENESYS[2]). His approach has the advantage that it addresses the problems of maintainability and consistency of the grammar (as used by both the generator and the parser), but it runs into problems of search space, and suffers from the limitation that the information is extracted from *artificial* random generation.

The current parser is an attempt to overcome the latter problem - but not at the expense of the former. The major emphases of the parser therefore can be stated as follows:

1. To maintain a close correspondence between the syntactic representation and the semantic representation which is to be extracted from it (this having implications for possible interleaved processing).

2. To obtain a syntactically and functionally rich parse tree (even when there is some ungrammaticality in the input).

3. To improve efficiency by (a) parsing incrementally and (b) guiding the parsing process by probability based prediction and the use of feature unification.

To this end we reject the strategy of adopting a PSG-type 'cover grammar', in the style of Kasper (1988) and adopt instead a systemic syntax as the basis of the parser. This is stored in the form of

1. Componence, filling and exponence tables, as described in section 2.3 and

2. The transition probabilities of the elements in the componence tables

The output of the parser's incremental evaluation of the parse can be exploited by a semantic interpreter of the kind described by O'Donoghue (1991a, 1993); see also Fawcett

[2]GENESYS is the main generator in the COMMUNAL project; It stands for GENErate SYStemically ;COMMUNAL stands for the Convivial Man Machine ... Using NAtural Language, and is a DRA sponsored project at the University of Wales College of Cardiff, UK.

(1993). Essentially, this runs the system networks in reverse to collect the features required[3].

In the rest of this paper we will introduce the concept of 'rich syntax' with respect to SFG (section 2), and then describe the techniques we adopt for parsing it (section 3). Finally, in section refconclusions we will evaluate the work done so far and discuss its limitations and future work envisaged.

2 Parsing for rich syntax

2.1 Systemic Functional Grammar

Before we describe the nature of systemic functional syntax, we need to point out that the syntactic structures (to be discussed in section 2.2) are not the heart of the grammar, but the outputs from the operation of the SYSTEM NETWORKS and their associated REALIZATION RULES[4].

SFG is a model of grammar developed from a functional view of language which has its roots in the work of Firth and the Prague School; Its major architect is Halliday. The more well known computer implementations of it have been developed mainly in the complementary field of Natural Language Generation (NLG). Some of these include Davy's PROTEUS(1978), Mann and Matthiessen's NIGEL(1985) and Fawcett and Tucker's GENESYS(1990). One significant NLU system based upon systemic syntax is Winograd's SHRDLU(1972).

The core of the grammar consists of a great many choice points, known as *systems*[5], at each which the system must take one path or another by choosing one of two (and sometimes more) semantic features. Quite large numbers of such systems combine, using a small set of AND and OR operators, to form a large network, as shown in figure 1. The big lexicogrammar which the parser described here is designed to work with has about 600 grammatically realized systems. The network is traversed from left to right, and each such traversal generates a 'selection expression' (i.e. a bundle) of features. Some of these have attached to them REALIZATION RULES, and it is these which, one by one, combine to build the semantically motivated 'syntax' structures that we shall describe in section 2.2.

For example, consider the fragment of a network shown in figure 1 below[6]. It shows a simplified version of the current network in the 'midi' version of the COMMUNAL grammar. What is not shown here is a detailed formal specification of the realization rules for the features collected by following the various possible pathways through the network. The first two realizations are however expressed informally: i.e. the meaning of [information] plus [giver] is realized by having the Subject (S) before the Operator (O), if there is an Operator, and if not by having the Subject before the Main Verb (M).

It should be noted that in the 'full' grammar there are probabilities attached to each system (or choice point). This enables the model to escape from the conceptual prison of the concept of grammaticality and enables us to account for very unlikely, yet possible choices being made.

[3] An alternative would be to have a separate compositional semantics component based on the functional paradigm described in this paper.

[4] Readers familiar with how a systemic functional grammar works may wish to skip this section.

[5] For these, and for an overview of the role of SFG in NLG, see Fawcett et al. (1993).

[6] For those familiar with SFL, there may be some interest in comparing the network given here with the traditional network for MOOD. It has been made much more explicitly semantic than the traditional MOOD network (which begins with [indicative] or [imperative], and then, if [indicative], either [declarative] or [interrogative]).

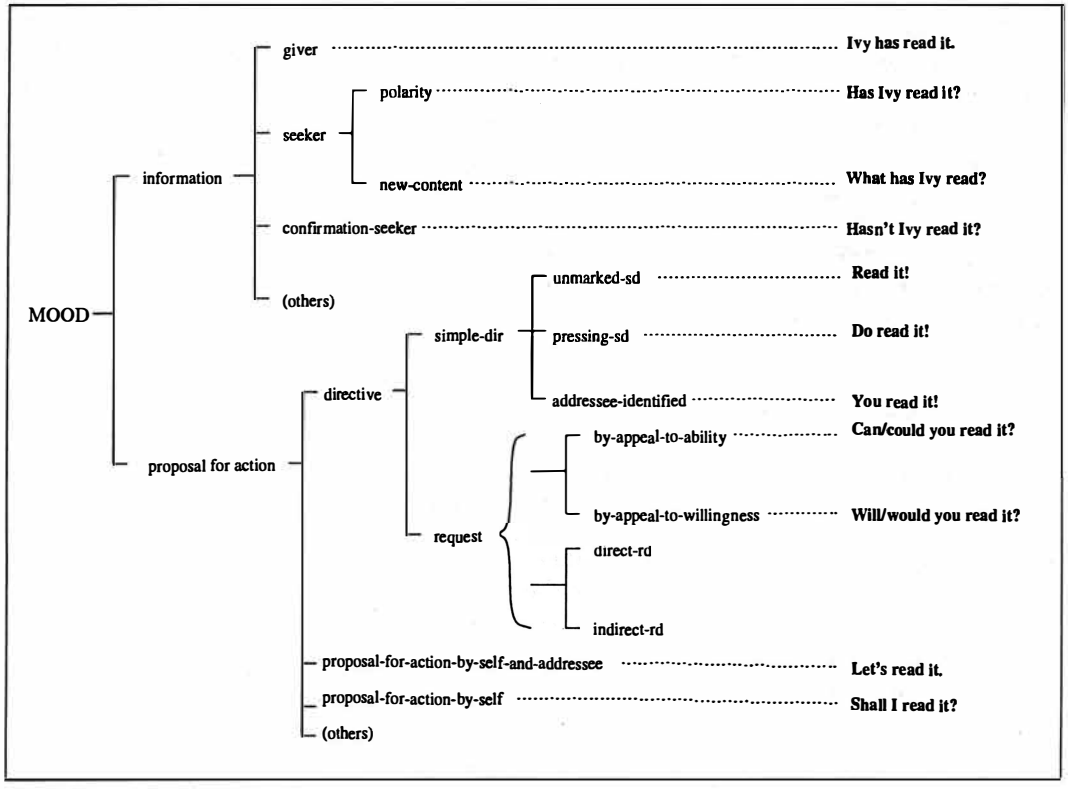

Figure 1: A simplified system network showing some of the meaning potential for MOOD in English (excluding much, e.g. POLARITY and realizations in tags and intonation)

2.2 Semantically motivated systemic functional aspects of the model

The syntactico-semantic analysis produced by the parser differs from traditional syntactic parse trees in at least the following four important ways.

1. Firstly, our model of 'syntax' distinguishes between the relationships of :

   (a) COMPONENCE, whereby a particular UNIT such as a nominal group (a 'full' noun phrase; denoted by 'ngp' in our systemic syntax) is composed of ELEMENTS (functional categories) *and*

   (b) FILLING, whereby such a UNIT 'fulfills', as it were, the functional role of the element it fills.

So, for instance, a ngp can have (among others) the components deictic determiner (dd), modifier (m) and head (h). At the same time it will 'fill' either the element functioning as Subject (S), a Complement (C1/2), a 'Completive' of a prepositional group, or some other element.

Together, the two concepts of filling and componence provide for (two types of) recursion in this model. Firstly, for example, a modifier in a nominal group can itself be filled by, among other things, a nominal group; this is the SFG approach to the phenomenon known (misleadingly) sometimes as 'noun-noun compounding', e.g. as in *their luxury flat sale*, where it is the ngp *luxury flat* (and not just the noun *flat*) that modifies *sale*.

The second type of recursion that is accounted for is COORDINATION. In cases such as *my brother and his wife (have arrived)*, both nominal groups (*my brother* and *and his wife*) fill the role of Agent (which is conflated with the Subject; see (c) below); they are jointly the Agent in the Process of *arriving*. Without the concept of filling, we would be forced to represent this relationship as if it were componence.

2. Secondly, the emphasis on *function* in the model is evident in the elements which constitute the final non-terminals in the syntax tree, which are categorised in terms of their *function* in the unit above, rather than as 'word classes'. In this scheme, the term 'noun' for example is a label for *one* of the classes of words which may expound the head of a ngp. Others may include pronouns, proper names or *one(s)*.

   Again, *very* is not treated simply as another 'adverb' (which misleadingly suggests that it functions similarly to *quickly*, etc), but as a 'temperer'. This is because it typically 'tempers' a quality of either a 'thing', as in (1b) below, or a 'situation', as in (1c). It is thus an element of what is here termed a 'quantity-quality group', in which the 'head' element, which expresses the 'quality', is termed the 'apex' (a) and the 'modifier' element, which 'tempers' it by expressing a 'quantity' of that quality, is termed a 'temperer'. This functional enrichment of the syntax provides a natural way to account for the difference between the functions of *very* and *big* in sentences 1a and 1b.

(1a) She noticed the big fat man.

(1b) She noticed the very fat man.

(1c) She ran very quickly to the window.

Finally, also note that, the grammar allows some (but not all) elements to be either 'expounded' by lexical or grammatical items or 'filled' by a syntactic unit. For example, consider the quantifying determiner (dq), which is EXPOUNDED in (2) and (3) by the ITEMS *a* and *one*, and FILLED by the nominal group (UNIT) *a bag* (dq h) in (4).

(2) He was a very interesting man.

(3) I'd like one cabbage, please.

(4) I'd like a bag of potatoes please.

Points 1 and 2 above, reflect and respect a specific commitment to maintaining the closest possible correlation between the units recognized in syntax and those recognized in semantics. Thus, in the unmarked case, a CLAUSE (cl) realizes a SITUATION (= roughly 'proposition') and a NOMINAL GROUP (ngp) realizes a THING ('object'). Hence our parser produces broad flat trees rather than those with multiple (often binary) branching; the 'work done' in models of the latter sort by the branching is done in our model by richer labelling. ie. Richer labelling reduces the need to represent relations by

distinctive tree structure variations, thus enabling us to restrict the branching to the definition of units that are semantically motivated. And this in turn greatly facilitates the transfer of the output from the parse to the next stage of processing.

3. Third, we also consider that the notion of ***absolute grammaticality***, which is intrinsic to phrase structure type grammars to be fundamentally misleading. Instead, we take a probabilistic approach to the question of what element may (or is unlikely to) follow what other element in a unit. The concept of ungrammaticality is thus simply one end of a continuum of probabilities from 0% to 100%. In this respect, our parser has characteristics in common with stochastic approaches to parsing, and so embodies, in effect, a *hybrid* model.

   Hence our parser accommodates some measure of ***ungrammaticality*** in the input text, and tries to extract ***whatever*** functional information it can from it – rather than rejecting it.Consider the example sentence below.

   (5) Car sales, in spite of the recession, *was* up by more than five per cent.

   The type of unification parser which enforces subject-verb agreement would simply return the verdict 'ungrammatical' on encountering such an utterance. Chart based parsers are a slight improvement, in that they would output the 'analysable' fragments of the sentence. Because our parser's goal is to return *some* semantically plausible interpretation, it returns a well formed parse tree out of such input[7].

4. Finally we should point out that our parser is helped to build its functionally rich output using the familiar concept of feature unification. Many other modern parsers use feature unification as a means of constraining the ever-growing parse forest caused by local ambiguity, while then passing up the 'unified features' for higher semantic processing. In a functionally oriented grammar such as SFG (in which the system networks from which the structures are generated are themselves entirely made up of semantic features) it is particularly natural to supplement the syntax tree through such 'percolated' features[8].

   Thus for instance, the features [manner], [place] and [time] are 'percolated' up from the items ***unexpectedly***, ***to Cardiff*** and ***on Friday*** respectively in (6).

   (6) He unexpectedly came to Cardiff on Friday.

2.3 The syntactic coverage of the parser

As will be evident from what has been said so far, there is no set of PSG-type re-write rules that specifies the syntax. Instead there are semantic features whose realization rules, collectively, build up syntactic structure. The information that a parser needs to have available to it is only implicitly present in the generator, and it is not in a form that is readily usable by the parser. O'Donoghue (1991b) explores one possible way of overcoming this problem, namely by extracting the 'rules' (or 'legal sequences') from sentences randomly generated by the generator (GENESYS). Our approach

[7] We take the view that such grammatical features are in fact not normally of any great help in disambiguation, and hence not of much use in further processing.

[8] Note that the use of features for constraining the parse forrest using for instance number agreement is not done here.

is to extract from the system networks and realization rules the information about syntax that is relevant to the work of the parser, and to state it in a form that is more amenable to this task[9].

The four major types of units recognized by the parser's syntax are the clause (cl), the nominal group (ngp), the prepositional group (pgp) and the quantity-quality group (qqgp)[10].

Of these,four units, the clause has by far the most complex and variable syntax. The elements of the ngp, pgp and qqgp on the other hand can be considered for practical purposes to be fixed, and the presence or absence of elements within such groups is reflected in our model in the transition probabilities (see section 3.1). Because of the fixed sequence of elements in these groups, we can at this point use a re-write rule notation to represent these.

ngp → dq, vq, dd, m, mth, h, qth, qsit
pgp → p, cv
qqgp → t, a, f [11]

Here, the '→' is used to denote the COMPONENCE relationship. Thus, for instance, a pgp can be composed of a preposition (p) and a completive (cv).

However, the above specifications have a number of grave limitations. They fail to show (1) those elements that are optional, (2) the degree of optionality, and (3) the dependencies that may hold between them, absolutely and relatively (e.g. there can be no vq if there is no dq, and it is highly unlikely that there will be a dq without an h). As we shall see, it is the information about probabilities that captures these facts.

The most complex of the groups, the clause, has a more variable potential structure which here we denote (for convenience) by the re-write rule[12]:

cl → B, A*, C2, O# ,S, O#, N, I, A*, X*, M, Cm, C1, C2, Cm, A*

As we shall shortly see, the information about adjacent elements expressed in these specifications, together with other vital information on optionality and probabilities, is made available to the parser in a somewhat different form.

A second type of information required by the parser is a set of statements about FILLING, i.e. about the elements which units can fill, thus [13] :

cl : A, C2, f
ngp : A, C1, C2, cv, mth, dq
pgp : A, C1, C2, qth, f
qqgp : m, A, C2, dq

Here, the ':' stands for the filling relationship. So, for example, the clause (cl) can fill an Adjunct, a second complement (C2) or a finisher (f).

To illustrate the type of syntax tree these two relationships together provide, consider

[9] We hope to be able to device a technique for automatic extraction of 'rules' from the system networks in future versions of the parser, but we defer this task for the present as it has been shown to be possible (O'Donoghue, 1991b).

[10] We should state here that the syntax described below handles only a subset of the 'midi grammar' contained in the system networks of GENESYS referred to above, and that we have nothing to say here about phenomena such as 'raising' and 'long-distance dependency' (though many aspects of discontinuity are already covered within GENESYS, and these types of phenomena are now being considered in the SFG framework).

[11] The key to the list of elements used in the parser is given in the Appendix.

[12] Here, * denotes recursive elements while # denotes that the Operator element can be 'conflated' with the functions of X (auxiliary) or M (main verb).

[13] Note that B, O, N, I, X and Cm are directly expounded by items.

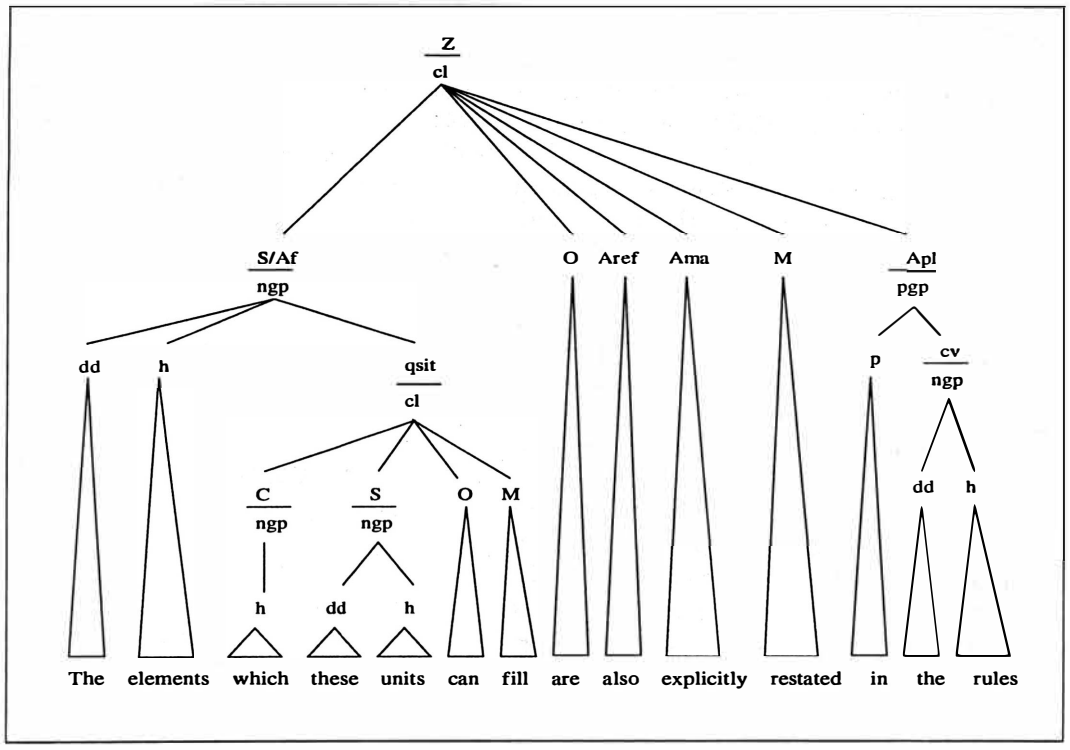

Figure 2: A Systemic Functional Analysis of a sentence

the typical SFG analysis of a sentence (Z) shown in figure 2 [14].

Note that in this analysis, the fragment *the elements which these units can fill* both corresponds to a single nominal group (filling the element of Subject and the participant role of Affected) and constitutes a single semantic 'thing' (or 'object').

We would argue, with Winograd (e.g. Winograd, 1972), that such 'flat' tree representations lend themselves more naturally to higher level processing than do trees with many branchings, because each layer of structure corresponds to a a semantic unit, and ultimately to a unit in the 'belief' representation.

[14] See appendix for 'conflation' abbreviations.

There is no genuine equivalent relationship to this in a PSG, because such grammars do not have the 'double labelling' of nodes in the tree as both element and unit (or, with coordination, units) described above. That is, there is no distinction between componence (unit down to element) and filling (element down to unit(s)). From the viewpoint of a parser, the relationship we are considering here is a unit-up-to-element table. Here the probabilistic information is extremely valuable; it is useful for the parser to know, for example, that it is relatively unusual for a clause to fill a Subject, but that a clause fairly frequently fills a Complement or Adjunct.

We have been considering the 'unit up to

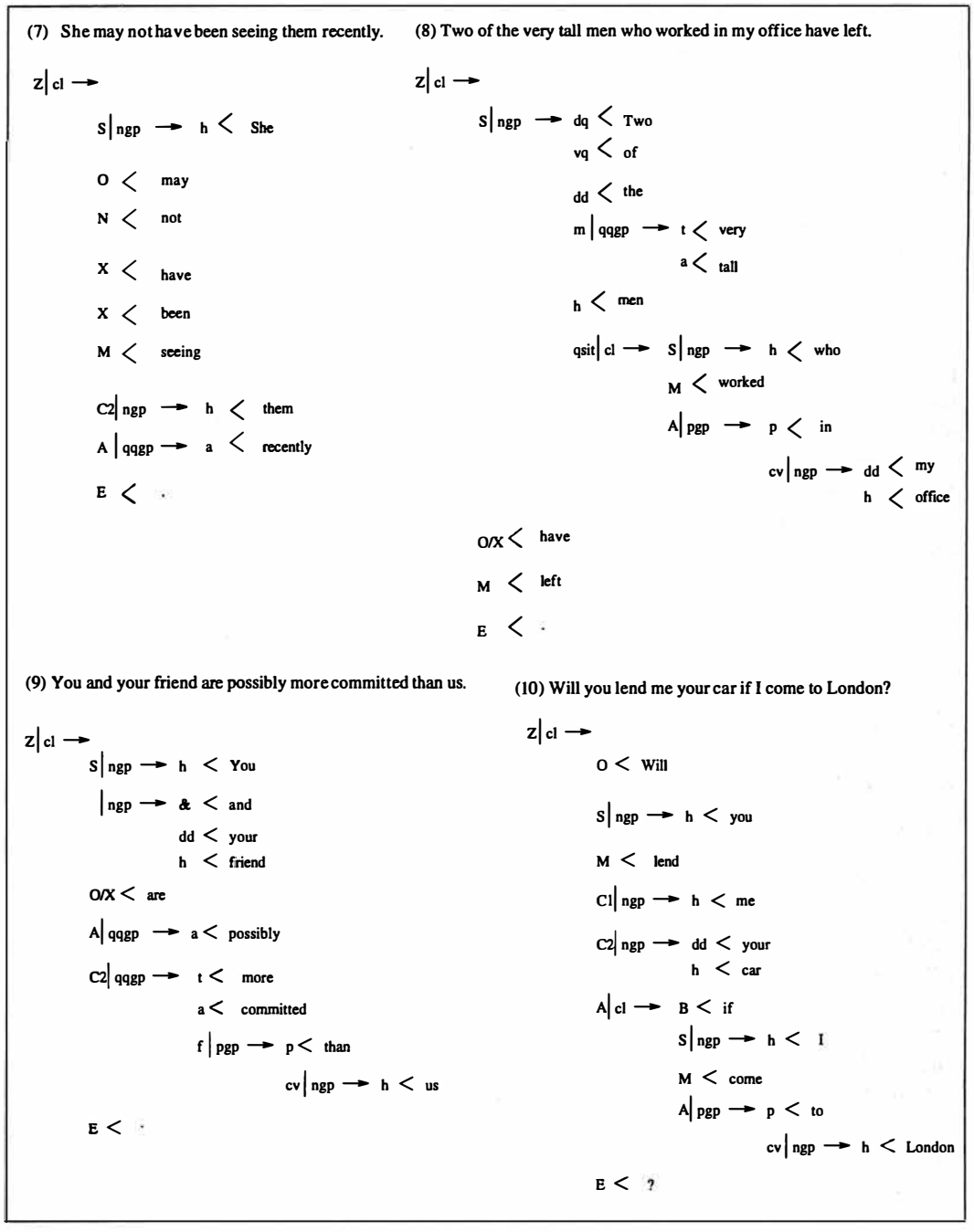

Figure 3: Some examples of the output of the parser

element' relationship of componence. Finally, there is the similar 'item up to element' relationship of EXPONENCE. This is a list of all the items (roughly, 'words') to be accepted by the parser , with the probabilities - for each sense of each word - of what elements each may expound (placed in order). The difference from the previous information source is that it is a very large and constantly modifiable component; the coverage of the unit up to element tables is in comparison quite limited (and less liable to revision in the light of successfully parsed new texts). This third component is therefore the equivalent in our parser of what is often termed the 'lexicon'.

As is shown by the example in figure 2, the SFG approach makes possible the output of syntactically rich, semantically oriented 'flat' tree parse structures. The typical outputs from the parser shown in figure 3 should, it is hoped, give a picture of the kind of data covered by the syntax, and so by the parser[15].

3 How the parser works

3.1 The basic algorithm

The fundamental strategy adopted in parsing for the rich functional syntax described in sections 2.2 and 2.3, is an adapted form of bottom up chart parsing with limited top down prediction. One of the main reasons for the adaptation of the chart parsing algorithm is to account for some of the context sensitivity exhibited by the SF syntax. For example, the possibility of an 'Operator' occurring after a Subject is dependent on it's non-occurrence before it. Similarly, certain types of Adjunct are mutually exclusive within a given clause. For this reason, our parser has lists of 'potential structure' templates (as shown in simplified form in section 2.3) instead of the usual CF-PSG type rules. These are augmented by the element transition probability tables and a probabilistic lexicon, to assist the adapted probabilistic chart parser implemented here.

Hence, the chart is composed of edges, each with a list of the elements that can 'potentially' occur following it, together with optionality and mutual exclusivity constraints, features associated with the edge and a unique probability score representing its likelihood of occurrence. As in the case of standard 'active' chart parsers, 'active' or hypothesis bearing edges too are maintained in a similar way.

The unification of edges is used only to perform a 'percollatory' function rather than a 'restrictive' one, so as to give less importance to the concept of 'grammaticality'. The aim of this is to allow some 'ungrammaticality' in order to extract at least *some* meaning from *any* utterance.

The probabilities themselves are calculated from three sources:

1. The item probabilities contained in the exponence table (the 'lexicon').
2. The filling probabilities for each unit.
3. The transition probabilities between elements within a unit.

In a given application of the 'fundamental rule', three component probabilities are used in working out a weighted geometric mean. It is our observation that, as Magerman and Marcus (1991) point out, joint probabilities calculated as products are not accurate estimates of such likelihoods owing to the events considered violating the independence assumption. The three probabilities thus affecting the new edge created are :

1. The probability of the 'active' edge in the 'attachment'.

[15]Note that at this stage the parser does not yet assign participant roles.

2. The probability of the 'inactive' edge of the 'attachment' .
3. The probability of the transition of elements involved in the 'attachment'.

For example, consider the fragment *the man* shown in figure 4.

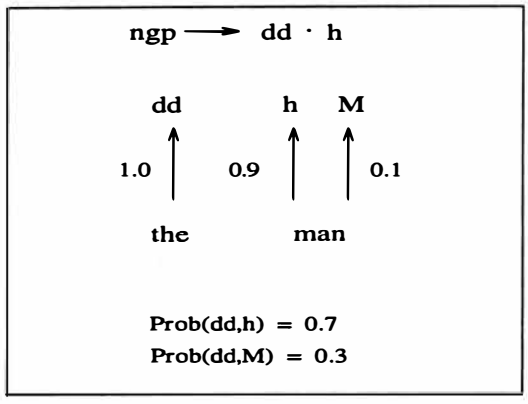

Figure 4: Edge creation with probabilities

In this situation, the two edges *the* and *man* would invoke the hypothesis (using the usual 'chart' notation):

ngp → dd . h

where the ngp is 'looking for' a head. In the ensuing unification of this active edge with h(man), we consider the probabilities of :

1. the active edge ngp(dd(the))
2. the inactive edge h(man) and
3. the transition (dd,h).

A geometric mean of the probabilities of 1 and 2 and a weighted 3 is attached to the new (inactive) edge ngp(dd(the), h(man)), that is thus formed. The weighting on the third component makes it favour the transition predictions over those of filling and exponence.

Subsequently, the filling probabilities of S, C, cv etc. will be considered. In the case where the above fragment begins the input string, the clause level transition probabilities will heavily favour the S to be the element being filled by the ngp (with a high score for the transition (\$,S)) than C or cv.

Consider the following example sentence to see how such a probabilistic model can assist in arriving at a correct analysis of a clause with lexical ambiguity :

(11) Did you notice him?

In this example, though *notice* could be both a head (noun) or a verb, the transition probability of head-head is very low. (Noun-noun compounding will not score well as the probability of *you* being able to fill a modifier is negligible). On the other hand, the transition probability of S-M is very high and so *notice* will be parsed as a M in the leading 'theories'.

Finally, consider the following 'garden path' type sentence to see how our probabilistic model copes with this type of ambiguity:

(12) The cast iron their clothes.

According to the COBUILD dictionary, *cast* is most commonly a noun(h) or a verb(M) while *iron* is most commonly a noun(h), but could also be a verb(M) and more rarely an adjective(a). A part-of-speech tagger encountering this input string will need to determine which of the transitions (dd,h,h), (dd,h,a), (dd,h,M), (dd,M,h) etc. are more likely. A tagger based on lexical co-occurrences or part-of-speech might well favour (dd,h,h), the former as it could have information about *cast* and *iron* being able to follow each other in this way and the latter to account for noun-noun compounding.

Our parser on the other hand, though initially favouring this theory like the class based tagger, will also advance the theory containing *iron* as a main verb(M). Once a certain

'height' of the parse tree is reached however, the probability score of theories treating *iron* as a noun(h) will diminish while those treating it as a verb(M) will be re-inforced by the high transition probabilities of the higher elements (S,M) and (M,C).

It is the availability of these 'higher' functional syntax level transition probabilities to the parser, that we suspect will enable our parser to perform better than (conventional) pure probabilistic part-of-speech level models.

3.2 The interactive interface

A major secondary goal of our work is to build a parser which can function as the front end to a complete interactive NLP system (COMMUNAL). To this end we have developed an interactive interface to the parser. Incoming items are tagged to focus the search space using a character reading input routine, which is responsible for providing (incrementally) the parser with a 'clean' input by :

1. Tagging punctuation according to the elements they expound.
2. Handling the syntax of large and decimal numbers.
3. Flagging abbreviations appropriately.
4. Signalling unknown words or assigning likely elements which might expound them

The 'final non-terminals' output by this routine are input to the parser incrementally, while simultaneously accepting further input. Thus by the time the user input is completed (by the tagger encountering an 'Ender' item) much of it has already been analysed by the parser.

The incremental nature of processing at this syntax level can be further exploited at higher 'interpretive' stages of analysis because of the well annotated 'flat' parse trees produced and their (near) one-to-one correspondence with semantic objects in the SFG adopted. (See Van der Linden (1992, p. 225) for reasons why traditional PSG-type grammars cannot in general be parsed incrementally in this way).

As an example, consider the following sentence.

(13) The boy with long hair saw Jill in the park.

Here, as soon as the user starts to input *Jill*, the item *saw* is tagged, with its syntactic context guiding the decision. Meanwhile, *The boy with the long hair* has already been identified as a nominal group (unit) with certain (quite limited) semantic properties, and it is thus ready for verification as, say, ⟨person102⟩ very early in the parse process.

4 Conclusions

4.1 Evaluation as a general functional parser

It is necessary firstly to evaluate our parser with respect to the richly annotated functional parse it produces. While time and space efficiency issues of the algorithm have not been brought to bear too heavily on the work done, the techniques adopted are general enough to be used for parsing other functional grammars represented as 'structural templates' (and supplemented by features and transition probabilities, and filling and exponence tables), with minimum modification to the algorithm itself.

The information contained in the flat parse trees constructed by the parser, while being richer in content, also allow for natural interleaving of syntax with higher semantic and pragmatic processing.

In this sense, we consider the current parser to be a successful precursor to a fully proba-

bilistic chart parser for functionally rich grammars.

More detailed formal evaluation, both of time-space efficiency of the algorithm and the parser's accuracy in analysing free text needs to be postponed for the present, until the parser is 'trained' on the fully systemically (hand) parsed Polytechnic of Wales (POW) corpus[16]. At the time of writing, a tool for the extraction of the necessary probabilities has been implemented (Day, 1993), though it needs as yet to be linked to the parser's probability module.

4.2 Evaluation as a front-end to COMMUNAL

Though the general algorithm is concerned with text parsing, our specific area of application is to use the parser as a front-end to the COMMUNAL NLP system, which is already equipped with a large systemic functional grammar embodied in its generator GENESYS. For this reason, the parser is equipped with an interactive interface which acts on input in an incremental way. It is also able to achieve a significant coverage of the 'midi' version of the GENESYS grammar. Our thesis is that this prototype parser will lend itself to being substantially extended to cover other complex grammatical phenomena handled by the 'full' version of the grammar, without the need to make any major alterations to the techniques employed in it.

4.3 Limitations and further work

One of the main limitations of the integrity of the system is that of the parser needing to be manually supplied with grammatical information embodied within the generator, GENESYS. An urgent need therefore is for a technique for extracting this information directly without human intervention. This would enable any grammar represented in system network notation to be compiled into a parsable form.

The main source of lexical probabilities for the parser has been West (1965), while element transition probabilities have been extracted (using the aforementioned interactive tool) from the POW corpus. For a more consistent approach non-reliant on human intervention, more work is needed on developing a non-interactive version of the parser which is able to train on hand parsed corpora.

The improvement of these aspects of the system will allow the current parser to be used as a robust 'real text' parser and to be incorporated into a NLU system capable of true interleaved processing.

[16]This is available from ICAME's archive at the Norwegian Computing Centre for the Humanities in Bergen, Norway.

Reference

Atwell, E. S., Souter, C. D., & O'Donoghue, T. F. (1988). Prototype parser 1. Tech. rep. 17, Computational Linguistics Unit, University of Wales College of Cardiff, UK.

Brew, C. (1991). Systemic classification and its efficiency. *Computational Linguistics*, *17*(4).

Davy, A. (1978). *Discourse production: A computer model of some aspects of a speaker*. Edinburgh University Press, Edinburgh, UK.

Day, M. D. (1993). The interactive corpus query facility: a tool for exploiting parsed natural language corpora. Master's thesis, University of Wales College of Cardiff, UK.

Fawcett, R. P. (1993). A generationist approach to grammar reversibility in natural language processing. In Strzalkowski, T. (Ed.), *Reversible Grammar in Natural Language Generation.* Dordrecht: Kluwer.

Fawcett, R. P., & Tucker, G. H. (1990). Demonstration of genesys: a very large, semantically based systemic functional grammar. In *The 13th International Conference on Computational Linguistics (COLING-90)*, pp. 47–49.

Fawcett, R. P., Tucker, G. H., & Lin, Y. Q. (1993). How a systemic functional grammar works: the role of realization in realization. In Horacek, H., & Zock, M. (Eds.), *New Concepts in Natural Language Generation: Planning, Realization and Systems.* Pinter, London.

Kasper, R. T. (1987). A unification method for disjunctive feature descriptions. In *Proceedings of the 25th Annual Meeting of the Association of Computational Linguistics.*

Kasper, R. T. (1988). An experimental parser for systemic grammars. In *The 12th International Conference on Computational Linguistics (COLING-88).*

Magerman, D. M., & Marcus, M. P. (1991). Pearl: A probabilistic chart parser. In *Procedings of the 2nd International Workshop on Parsing Technologies.*

Mann, W. C., & Matthiessen, C. (1985). Nigel: A systemic grammar for text generation. In Freedle, R. O. (Ed.), *Systemic Perspectives on Discourse.* Ablex.

Matthiessen, C. M. I. M. (1991). *Test generation and systemic functional linguistics.* Pinter, London.

Mellish, C. S. (1988). Implementing systemic classification by unification. *Computational Linguistics, 14*(1).

O'Donoghue, T. F. (1991a). A semantic interpreter for systemic grammars. In *Proceedings of the ACL Workshop on Reversible Grammars.*

O'Donoghue, T. F. (1991b). The vertical strip parser : a lazy approach to parsing. Report 91.15, School of Computer Studies, University of Leeds, UK.

O'Donoghue, T. F. (1993). Semantic interpretation in a systemic grammar. In Strzalkowski, T. (Ed.), *Reversible Grammar in Natural Language Generation.* Dordrecht: Kluwer.

Patten, T. (1988). *Systemic Text Generation as Problem Solving.* Cambridge University Press.

Patten, T., & Ritchie, G. (1986). A formal model of systemic grammar. Research paper 290, Deptartment of AI, Edinbutgh University, UK.

Van der Linden, E.-J. (1992). Incremental processing and the hierarchical lexicon. *Computational Linguistics, 18*(2).

West, M. (1965). *A general service list of english words.* Longmans.

Winograd, T. (1972). *Understanding Natural Language.* Academic Press Inc.

Appendix

Symbol	Name	Also Known As	Function
Clause elements:			
cl	Clause	Sentence	Realizes a 'situation'
&	Linker	Conjunct(coord)	Links two 'equal' units
B	Binder	Conjunct(subord)	Binds subordinate unit into a higher unit
A	Adjunct	Adverbial/Prepositional phrase	Realizes circumstantial roles, etc. in the clause
C1/2	Complements	Object(direct/indirect)	Realizes main participant roles in the clause (together with S)
O	Operator	First Auxiliary	Realizes mood, negation, emphasis or polarity, tense
X	Auxiliary	Auxiliary	Realizes tense, aspect, passives
O/M	Operator/Main Verb	Modal Verb	Operator functioning as the Main Verb of the clause
S	Subject	First NP	Specifies the subject role of the clause
N	Negator	Negator	Negates clause
I	Infinitive	Infinitive	Used in infinitive clauses
M	Main Verb	Verb	Specifies the process, constrains the roles in clause and tense
Cm	Main-Verb-completing complement	Particle	Completes the meaning of the Main Verb
Nominal group elements:			
ngp	Nominal group	Noun phrase	Realizes 'things'
dq	Quantifying determiner	Determiner	Quantifies the nominal group
vq	'of' element	Preposition	Shows 'selection' relationship
dd	Deictic determiner	Determiner	Marks definiteness in the nominal group
m	Modifier	Adjective	Modifies the 'head' of the group

mth	Thing modifier	Noun modifier	Modifies the 'head' of the group
h	Head	Noun	Marks the head noun or is a pronoun or proper name
qth	Qualifier	PP	Modifies the 'head' by a prepositional group
qsit	Qualifier	Relative clause	Modifies the 'head' by a clause

Prepositional group elements:

pgp	Prepositional group	PP	Realizes a 'minimal relationship' plus a thing
p	Preposition	Preposition	Expresses minimal relationship
cv	Completive	NP in PP	Expresses the thing

Quantity-quality group elements:

qqgp	Quantity-quality gp.	Adverbial or Adjectival	Realizes (quantities of) qualities
t	Temperer	Intensifier	Tempers the quality of the following adverb/adjective
a	Apex	Adjective or Adverb	The 'head' of the group
f	Finisher	–	Completes meaning of temperer

Participant roles: (Conflated with S and C1, C2)

Af	Affected	Patient
Ag	Agent	Actor
At	Attribute	
Ca	Carrier	
Cog	Cognizant	
Cre	Created	
Em	Emoter	
Loc	Location	
Perc	Perceiver	
Ph	Phenomenon	
Pos	Possessed	

Adjuncts bearing circumstantial roles:

Afreq	Frequency
Ahyp	Hypothetical
Ama	Manner
Apl	Place
Apol	Politeness
Areas	Reason
Atp	Time position
Ausu	Usuality