Partial Descriptions and Systemic Grammar

Chris Brew
Centre for Cognitive Science
University of Edinburgh
2 Buccleuch Place
Edinburgh
U.K.
chrisbr@uk.ac.ed.cogsci

Abstract

This paper examines the properties of feature-based partial descriptions built on top of Halliday's systemic networks. We show that the crucial operation of consistency checking for such descriptions is NP-complete, and therefore probably intractable, but proceed to develop algorithms which can sometimes alleviate the unpleasant consequences of this intractability.

1 Introduction

Halliday's system networks [3] lay out in diagrammatic form the interlinked sets of linguistic choices which a speaker must make in order to generate an utterance. As such they have formed the basis of several computer models of natural language generation [4,11].However, as Mellish [12] has pointed out, a network can also be read as encoding a set of background constraints which restrict the co- occurrence of descriptive features, and hence as a specification of the way in which partial descriptions of linguistic objects can be combined. Although it is easy to combine feature sets, it is not always clear whether the resulting combined description can actually describe any well-formed linguistic object. Thus the main task we face is that of checking feature sets for consistency.

Consider for example the framework given by Winograd for the description of English pronouns, which is reproduced in figure 1. Suppose that a natural language system has somehow recovered the information that a pronoun in a particular position can be treated as both *third* (person) and *subjective*. At this stage we could be dealing with either *"they"*, *"he"*, *"she"* or *"it"*. Were we to combine this underspecified pronoun with the further description *feminine* we should know for sure that the pronoun described is *"she"* and the number has to be *singular*, since the network dictates, in a way which will be explained in detail below, that the choice between *feminine*, *masculine*, and *neuter* is only applicable to third person singular pronouns. The network thus provides the raw material for particular sorts of limited inference about the behaviour of pronouns. We wanted to investigate the mathematical properties of systemic networks in order to better understand the nature of the constraints on feature structure which they are capable of expressing.

Both Mellish and Kasper [7] provide translations of systemic networks into non-graphical formalisms: Mellish expresses constraints as axioms within a simple subset of predicate logic, while Kasper uses an extended version of Functional Unification Grammar [10]. Unfortunately the methods which they then use to check for consistency are powerful general methods which may incur considerable computational cost.

We initially hoped to show that systemic networks constitute a low-power constraint language which combines the twin goals of linguistic credibility and computational tractability. While the main result we present in this paper is a negative one indicating the unexpected power of systemic networks, we do go on to present suggestions about how networks can be exploited in natural language applications.

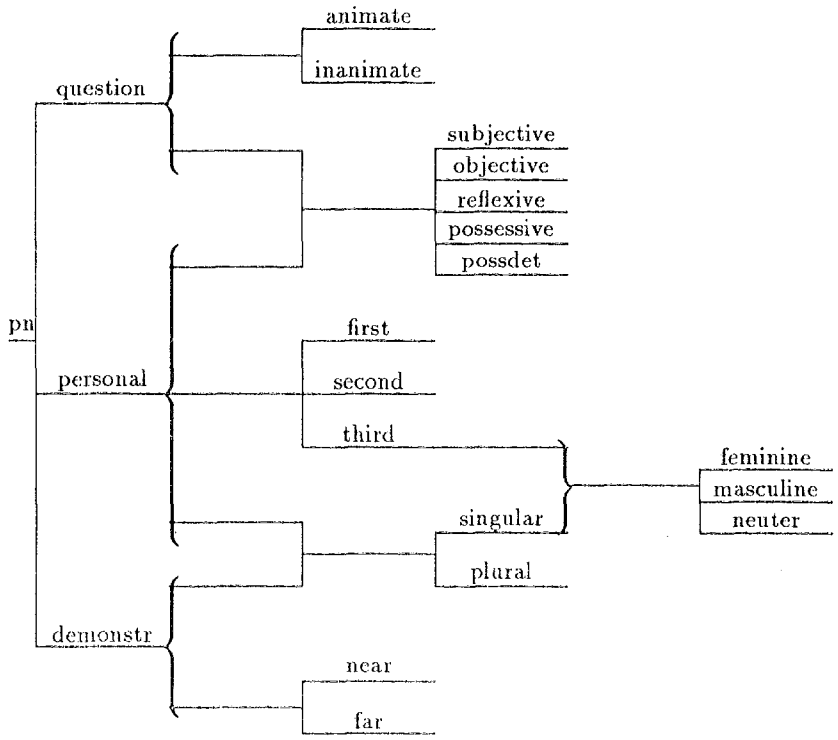

Figure 1: A pronoun network for English

2 What's in a net?

Our first task is to provide a precise characterisation of the information expressed by a systemic network. We begin by defining a way of labelling systemic networks, then provide a translation which maps labelled networks into collections of axioms expressed in the form of propositional logic. This work is a slight refinement of a very similar approach used by Mellish.

Figure 1 contains examples of each of the four types of system which we need to consider, linked together in such a way as to produce a description of the possible forms of English pronouns.

The leftmost system is a *choice* system expressing the opposition between *question*, *personal* and *demonstrative* pronouns. Within these broad classification further distinctions operate. For example *question* pronouns may be *animate* or *inanimate*, and must also make a choice between various cases. The system which expresses the necessity of making two simultaneous choices is indicated with a left curly bracket, and is the *and* system. Note that there are two routes to the choice of case, one from *question* and the other from *personal*. The system which ties these two routes together is called the *disjunctive* system. Finally, the rightmost system is a choice between various grammatical genders. This system can only be reached if a pronoun is both *third* and *singular*. The system involving the right-hand curly bracket which expresses this is called the *conjunctive* system.

3 Labellings for networks

We now establish technical definitions of two types of labelling for systemic networks.

3.1 Basic Labellings

A *basic labelling* is defined to be a partial function from lines to names such that

- A line receives a name if and only if there is a **choice** system to whose right hand side it is directly attached.
- No two lines carry the same name.

Figure 1 shows a basic labelling

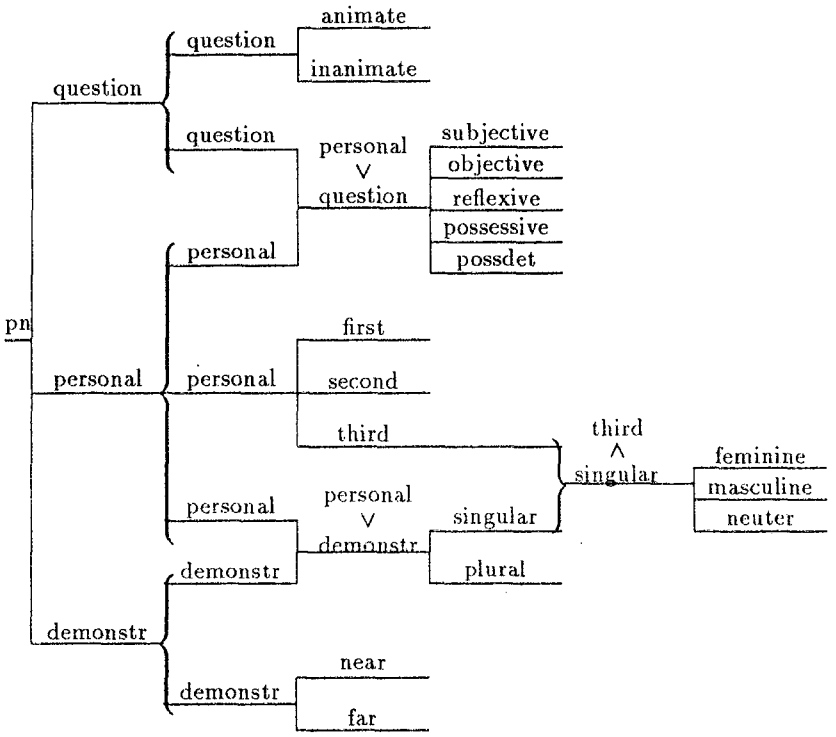

Figure 2: An exhaustively labelled version of Winograd's network

3.2 Exhaustive labellings

Let $\mathcal{F}$ be a 1-1 function from lines to names for these lines. $\mathcal{F}$ is an **exhaustive labelling** of a network if and only if the following conditions hold:-

1. That part of $\mathcal{F}$ which provides names for lines attached directly to the right of **choice** systems must be a basic labelling.

2. If $\mathcal{F}$ assigns a line name l_{lhs} to the line which is directly attached to the left hand side of an **and** system, then it must also assign that name to the lines which are directly attached to the right hand side of that system.

3. If $\mathcal{F}$ assigns line names $l_1, l_2, \ldots l_m$ to the lines entering a **conjunctive** system (where l_1 is the label for the line appearing at the top of the system and l_m that at the bottom), then it must assign the label $l_1 \wedge l_2 \wedge \ldots l_m$ to the line which leaves that system

4. If $\mathcal{F}$ assigns line names $l_1, l_2, \ldots l_m$ to the lines entering a **disjunctive** system (adopting the same ordering convention as above), then it must assign the label $l_1 \vee l_2 \vee \ldots l_m$ to the line which leaves that system.

Figure 2 is an exhaustively labelled version of figure 1.

4 Translating labelled networks

We use a translation scheme given by Mellish [12] to produce a set of logical axioms equivalent in meaning to the original network. Mellish's scheme can be applied to all four types of system and the correct results will be produced, but with our labelling scheme only *choice* systems contribute to the meaning of the network. Each *choice* system translates into two axioms: an *accessibility* axiom, expressing the constraint that none of the labels to the right of the system can be selected unless the entry point of the system has been reached; and an *exclusivity* axiom expressing the fact that these labels are mutually exclusive.

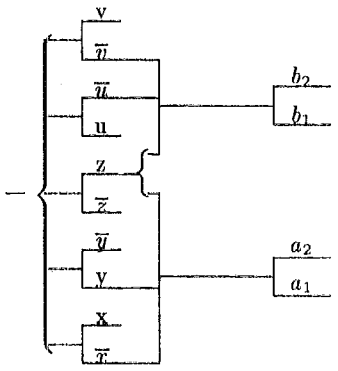

Figure 3: A network for 3SAT

5 Systemic classification is NP-hard

In this section we show that the problem of systemic classification is at least as hard as problems known to be NP-hard. This is done by constructing a polynomial time mapping Π from instances of the NP-hard problem called 3SAT to networks which can be tricked into solving this problem for us. For an introduction to similar linguistic applications of complexity theory see Barton et al [1].

If there were a polynomial time algorithm for checking arbitrary systemic networks, it would follow that 3SAT could be solved by the composition of the mapping that constructs the network with the algorithm that checks the network. Since this composition is itself a polynomial time algorithm we would then have a polynomial time solution for 3SAT, and hence for all other problems of the complexity class $\mathcal{NP}$. Thus the successful construction of Π implies that systemic classification is itself NP-hard.

5.1 The 3SAT problem

3SAT is the problem of determining the satisfiability of a boolean formula, stated in conjunctive normal form, in which exactly three variables occur in each clause of the conjunction. These variables may either be positive or negated, and may be repeated from clause to clause. It is known that 3SAT is just as hard as the problem of satisfiability for general boolean formulae (Barton et al provide a demonstration of this fact on pp 52-55 of [1]).

5.2 The mapping from 3SAT instances to networks

The mapping Π takes a 3SAT instance and produces a network. Let the name of the 3SAT instance be E and its length N_E.

- Make a list of the variable names used in E, counting positive and negative occurrences of a variable as the same. This can certainly be done in time polynomial in N_E using a standard sorting algorithm such as merge sort. Let the name of the list of variable names be V and its length N_V. We use the example of the very simple expression

$$(x \vee y \vee z) \wedge (z \vee \overline{u} \vee \overline{v}) \qquad (1)$$

- Construct a network consisting of a large **and** system feeding N_V parallel binary **choice** systems. Each **choice** system carries two labels, one corresponding to a variable name in V and the other formed by negating the label on the other branch of the system. The choice of prefix should be such that all labels on the resulting network are unique. This part of the process is polynomial in the length of V.

- For every clause in E, add a ternary **disjunctive** system linking the lines of the network having the labels corresponding to the three symbols of the clause. This part of the process involves scanning down the N_V systems of the network once for each clause of E, and is therefore also polynomial in N_E.

- Finally, binary **choice** systems are attached to the outputs of all the **disjunctive** systems introduced in the last stage. These systems are labelled with generated labels distinct from those already used in the network. This step is clearly also polynomial in N_E, requiring the creation of a number of **choice** systems equal to the number of clauses in E.

The network given in figure 3 is the one which would be produced from E. In order to use the constructed network to solve the satisfiability problem for E, we check an expression corresponding to the conjunction of all the three member clauses in E. This is built by choosing an arbitrary label from each of the rightmost **choice** systems. The conjunction of these labels is a consistent description whenever all the

clauses of E can be satisfied by the same value assignment. The **choice** systems to the left of the disjunction express the facts that no variable can be simultaneously true and false. A correct checking algorithm will succeed in just those circumstances where there is at least one value assignment for the variables of E which makes E come out true. Systemic classification is therefore at least as hard as the other problems in $\mathcal{NP}$,and we should be very surprised to find that it can in general be solved in polynomial time.

6 Checking systemic descriptions

Although accurate checking of systemic descriptions is an NP-hard problem, it is still possible to devise algorithms which carry out part of the process of checking without incurring the cost of complete correctness. Our algorithm depends on a pre-processing step in which the original network is split into two components, each of which embodies some but not all of the information that was present at the outset.

The first component is a simplified version of the original network, in which no *disjunctive* systems are present. This is achieved by removing all disjunctive systems, then re-attaching any dangling systems to a suitable point to the left of the position of the disjunction. For convenience in book-keeping we introduce special generated features which take the place of the disjunctive expressions that appear in the labelling of the original network. Figure 5 shows the result of peeling awaythe disjunction in figure 4.

The second component of the network consists of a collection of statements indicating ways in which the generated features may be discharged. For the example network we would have had to note that

$$genfeat \equiv c_1 \vee c_2$$

Taken together the simplified version of the network and the statements about generated features contain all the information needed. The simplified network is now amenable to deterministic and efficient checking procedures, including reductions to term unification as proposed by Mellish. The efficiency of these techniques hinges on the removal of *disjunctive* systems.

The second stage of checking involves the search for a consistent way of discharging all the

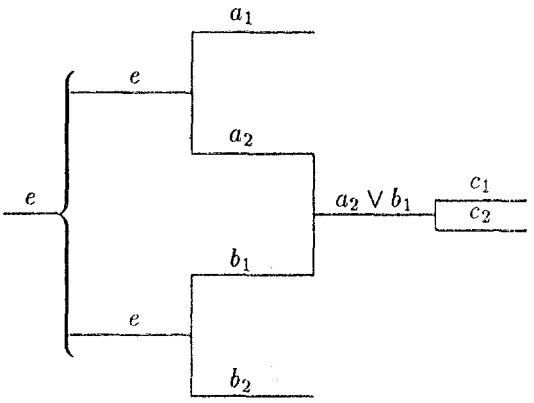

Figure 4: A small example network

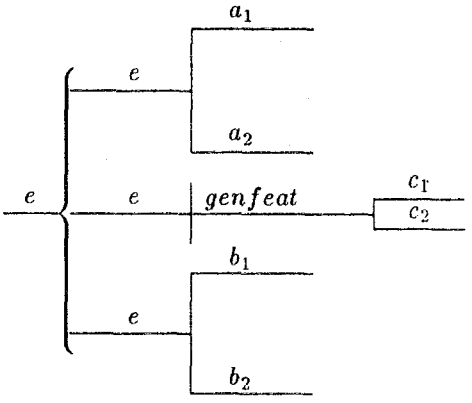

Figure 5: The transformed version of the example

generated features introduced by the first stage. This is the potentially costly part of the checking process, since separate disjunctions may conspire to produce exponentially many different alternatives which have to be checked. It was to be expected that the process of systemic checking would involve an exponential cost somewhere, so this is no surprise.

Even the second stage of checking is cheap unless two separate conditions hold

1. The description produced by the first stage of checking must involve many generated features.

2. The generated features must be interdependent, in that the way in which one feature is discharged constrains the way in which other features can be discharged.

We can't be sure whether the first condition is going to hold until we see the output of the first stage, but we can estimate the extent to which features interact by inspecting the checking rules which arise when the network is partitioned. Thus, while we can't promise that the use of systemic networks will ensure tractability for arbitrary grammars, we can help linguists to catch potential problems in the formulation of their feature systems during grammar development, and avoid the risk of unexpected combinatorial explosions during the exploitation of the grammars in question.

References

[1] G.Edward Barton, Robert C.Berwick, and Eric Sven Ristad. *Computational Complexity and Natural Language*. MIT Press, 1988.

[2] Andreas Eisele and Jochen Dörre. Unification of Disjunctive Feature Descriptions In Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics, Stanford University, Stanford, CA, July 6-9, 1987.

[3] M.A.K Halliday. The form of a functional grammar. In G.R.Kress, editor, *Halliday:System and Function in Language*, chapter 2, pages 7–25. Oxford University Press, 1976.

[4] George Houghton. *The production of Language in Dialogue: A computational model.* PhD thesis, University of Sussex, 1986.

[5] Lauri Karttunen. Features and Values Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics, Stanford University, Stanford, CA, July 2-6, 1984.

[6] Robert T. Kasper. Feature Structures: A Logical Theory with Application to Language Analysis PhD thesis, University of Michigan, 1987.

[7] Robert T. Kasper. An Experimental Parser for Systemic Grammars In Proceedings of the 12th International Conference on Computational Linguistics: COLING 88, Budapest: August 1988.

[8] Robert T.Kasper. A Unification Method for Disjunctive Feature Descriptions In Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics, Stanford University, Stanford, CA, July 6-9, 1987.

[9] Robert T.Kasper, and William Rounds. A Logical Semantics For Feature Structures In Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics, Columbia University, New York, NY, June 10-13, 1986.

[10] Martin Kay. Parsing in functional unification grammar. In D.R. Dowty, L. Kartunnen, and A.Zwicky, editors, *Natural Language Parsing*, pages 251–278. Cambridge University Press, Cambridge, England, 1982.

[11] W.C.Mann and C.Mathiessen. Nigel : A systemic
grammar for text generation R.Benson and J.Greaves *Systemic Perspectives on Discourse*, Ablex,London,England,1985.

[12] C.S. Mellish. Implementing systemic classification by unification. *Computational Linguistics*, 14(1):40–51, 1988. Winter.

[13] Terry Winograd. *Understanding Natural Language*. Academic Press, 1972.