INCREMENTAL INTERPRETATION AND COMBINATORY CATEGORIAL GRAMMAR

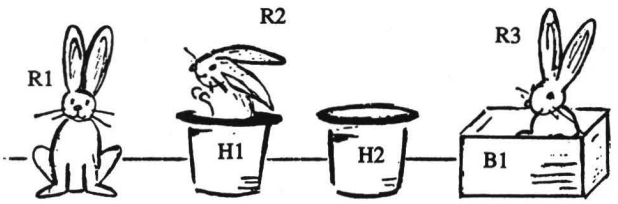
Nicholas J. Haddock
Department of Artificial Intelligence
and Centre for Cognitive Science,
University of Edinburgh,
Edinburgh EH8 9LW, U.K.

ABSTRACT

The paper is organised around a particular problem of anaphoric definite reference. A solution to this problem requires sentence processing which allows semantic representations to be evaluated *incrementally*, as a phrase is read from left to right. The paper shows how a proposal made by Mellish (1985), to regard incremental semantic evaluation as a constraint satisfaction task, can provide a natural solution to the problem of reference. The incorporation of Combinatory Categorial Grammar allows a straightforward relationship to be stated between the incrementally assembled syntactic and semantic representations.

I INTRODUCTION

The concern of this paper is to show how a computational model of incremental interpretation suggests a solution to a particular problem of definite reference. Suppose we have a context in which there are three rabbits (R1, R2, and R3), two hats (H1 and H2) and one box (B1), where one rabbit is in a hat and one is in the box:



If both speaker and hearer are aware of this context, the second rabbit, R2, can be referred to by means of the following complex NP:

(1) the rabbit in the hat

This expression can be uttered without any previous talk of rabbits or hats. What is interesting about it is the use of the definite determiner in the inner, simple NP, *the hat*. As a whole, the expression in (1) sounds perfectly natural, and so suggests that a unique hat is identifiable to the hearer. But given that there are two hats, and not one, in the scene, why is the definite permissible? Any compositional accounts of NP semantics, including those of Winograd (1972) and Hirst (1983), would judge (1) to be infelicitous. Such theories would require the somewhat convoluted version in (2), if a definite is to be used at all to specify the hat.

(2) the rabbit in the hat with a rabbit in it

A semantics organised around the criterion of incremental interpretation should have less trouble with (1). If we assume that a hearer *incrementally* evaluates a semantic representation — after each word, say — the empty hat in the scene will never really be considered a viable candidate for the inner NP. When the word *rabbit* is reached, a hearer can collect together in his mind the set of rabbits in the context. After the preposition, this set can be refined to contain only rabbits which are *in* something and, most importantly, the hearer can start thinking about another set of objects, those which have rabbits in them. There is only one hat in this new set and so by the time the inner NP is processed a definite determiner sounds natural. How can we pin this idea down in more explicit, computational terms?

II REFERENCE AND CONSTRAINT SATISFACTION

Mellish (1985) has proposed that the general process of incremental reference evaluation, sketched above, can be directly expressed by a constraint satisfaction algorithm of the kind presented by Mackworth (1977). Consider the following predicates, which may arise from the fragment *the rabbit in* (ignoring the determiner):

(3) rabbit(e1)
in(e1,e2)

Constraint satisfaction involves determining, for each of the variables e1 and e2, the entities in the discourse which satisfy the predicates, and collecting these into candidate sets for each variable. (We shall refer to these variables as *extension variables.*) Given the context provided for (1) above, this operation will label e1 with the set {R2,R3} and e2 with {H1,B1}. The advantage of the scheme is at once clear: satisfying the constraints in (3) determines a candidate set for e2 as well as e1, and so the reference of any NP which follows the fragment is already partially-evaluated. This offers a natural solution to the problem of non-compositionality exhibited in (1); a solution which is difficult to obtain in compositional semantic systems.

The following treatment of our problematic phrase illustrates two significant improvements on Mellish's own account of anaphoric definite reference. First, the phrase is evaluated strictly word-by-word, in left-to-right order; there is linguistic, psychological and intuitive support for this general position (Steedman (1987)). In contrast, Mellish's system would evaluate the two nouns before considering the preposition which relates them. Second, the relationship between syntactic and semantic descriptions is made explicit. Mellish himself points out that his system "lacked precision" on this issue (Mellish 1985:114).

III THE MODEL

The model sees syntax as controlling the way in which semantic representations are combined. And the goal is to pro-

vide an incremental semantics, so that a phrase receives its interpretation in stages, after each word is read. If this process is to accord with the rules of syntax, then the grammar too must be incremental.

A framework which might be helpful in this respect is Combinatory Categorial Grammar (Ades and Steedman (1982), Steedman (1985)). Like all categorial grammars, Combinatory Grammar employs a rule of *function application* to combine functions and arguments:

(4) $X/Y + Y \rightarrow X$

Steedman adds further combinatory rules, including *function composition:*

(5) $X/Y + Y/Z \rightarrow X/Z$

As we shall see, function composition is especially relevant to the issue of incrementation because it allows "incomplete" constituents, such as *the rabbit in* or *you can believe that I like,* to be elegantly described.

How can a constraint-based semantics be tailored to a system of combinatory grammar? Suppose we associate a constraint with each nominal word and an extension variable with its category. The noun *box* could have the following entry in the lexicon:

(6) box := N_{e7} : box(e7)

Box is defined (":=") as being an N linked to the variable e7. The identity of this variable is arbitrary; all that matters is that each word in the string is given a novel extension variable, to avoid unintended name clashes. The definition above associates a constraint with the noun, indicating that any value assumed by e7 must name a box in the context. Function categories differ only in that they are coupled to *mappings* between variables. Consider the category for a determiner, NP/N: given a noun to the right, this yields an NP. With semantics attached, the definite determiner appears as:

(7) the := NP_{e1}/N_{e1} : unique(e1)

This simply states that the variable associated with the noun is the same as the one associated with the NP, e1. This ensures that the constraint arising from the determiner, *unique(e1),* will restrict the set of candidates that apply to the noun, whatever they turn out to be.

The combination rules of the grammar have responsibility for relating these representations to each other. In relating the above determiner and noun, for instance, the variables e1 and e7 should be identified, because they are really both about the same set of candidate entities. Thus, in addition to the usual cancellation in the syntax, the rules of application and composition must *equate* extension variables. This is achieved by imposing a constraint of equality on the variables concerned, and the application of *the* to *box* appears as the reduction in (8).

(8) $NP_{e1}/N_{e1} + N_{e7} \rightarrow NP_{e1}$: equal(e1,e7)

The constraint of equality expresses the fact that the candidates for e1 must be the same as those for e7; all constraints which apply to e1 must also apply to e7, and vice versa.

Let us now go back to the phrase in (1), and consider the representations bestowed upon its component parts. Prepositions are functions taking an NP argument on their right to form another function, which must modify a noun on the left. They are therefore categorised as (N\N)/NP, where the leftward application of the inner function is indicated by a backward slash '\'. Because of its relational standing, *in* maps between two distinct extension variables:

(9) in := $(N_{e3}\backslash N_{e3})/NP_{e4}$: in(e3,e4)

Thus, e3 will be identified with the variable of the head noun and related to e4, which represents the prepositional object.

Any noun is form-class ambiguous between its primitive category, N, and a *type-raised* category, N/(N\N), a function over noun-modifying functions. Type-raising gives a noun like *rabbit* the opportunity to compose with *in,* categorised as (N\N)/NP. Note that this operation simply alters the noun's syntactic status, and does not complicate the semantics:

(10) rabbit := $N_{e2}/(N_{e2}\backslash N_{e2})$: rabbit(e2)

The means by which the parser decides between the two noun categories is beyond the scope of the present paper; suffice it to say that the choice is made in accordance with "The Principle of Referential Failure" of Altmann (1987). In the following analysis of (1), we will assume the type-raised version of *rabbit,* which expects modifying material, and the primitive version of *hat,* which closes the whole NP.

The constraint arising from a definite article, introduced above as *unique(e1),* is different in kind to the constraints arising from *rabbit, hat* and *in.* It is defined as follows:

(11) unique(eN) is true iff the candidate set for eN is singleton

Whereas a constraint like *rabbit(eN)* is a predicate over the particular discourse entities which make up the set for eN, *unique(eN)* is a predicate over the set as a whole. As such, *unique(eN)* is a "meta-constraint": it makes a check on the degree to which a given candidate set is constrained. There is another aspect of (11) which distinguishes it from the rest. Whereas *rabbit(eN)* has its effect as soon as the word *rabbit* is read, the constraint in (11) is only enforced when the NP corresponding to the variable eN is syntactically closed.

Finally, a note on parsing. The input string is processed by a non-deterministic variant of a shift-reduce parser, working from left to right in the string and reducing whenever it can. The details of the algorithm are omitted from the following trace, and the interested reader is referred to Pareschi and Steedman (1987), and Haddock *et al.* (1987) for further discussion of parsing issues.

IV *THE RABBIT IN THE HAT*

The table in (12) indicates the successive states of the system as it reads each of the five words in (1) and makes a combination with the previous constituent.

The first move in processing the phrase is to read the category of the initial determiner, NP_{e1}/N_{e1}, and add its condition to the constraint satisfaction process (CSP). This simply asserts that e1 is expected to have a single candidate, when the corresponding NP is closed. (Notice that the constraint is labelled with an asterisk in (12a), because it applies only at the time of syntactic closure.) The second word is now read, as defined in (10). This places a condition of rabbitness on the variable e2, which at present is unconnected to e1. The addition of *rabbit(e2)* prompts the CSP to search for suitable values for e2 in the context, and it finds three entities fitting the description: R1, R2 and R3. The two categories can now be combined, using composition. This cancels N_{e1} and N_{e2}, producing the category in (12b). A constraint of equality arises from the reduction, meaning that e1 and e2 are treated as alternative names for the same set.

The preposition (defined in (9)) is now processed, introducing two new variables to the CSP. Satisfying the relation there-

fore produces two new sets of candidates: one for all containing objects (e4) and one for all objects contained (e3). The subsequent composition with the category in (12b) matches e3 with e2. This removes the first rabbit, R1, from the set for e1 (which equals e2) as R1 is not contained in anything. (12c) records the current category and present state of the CSP. Notice that the second hat, H2, is not included in the set of candidates from which the inner NP must later take its reference, because it does not contain R2 or R3.

The second determiner is read, with the category NP_{e5}/N_{e5}. Reducing *the-rabbit-in* with *the* does nothing to further refine the set of containing objects, but the system now expects this particular set to be a singleton when the phrase is complete. As (12d) indicates, the incremental evaluation of *the rabbit in the* has created two distinct sets of candidates, for the two NPs in the phrase. The variables e1, e2 and e3 all denote the set {R2,R3}, holding possible referents for the complex NP, while e4 and e5 specify {H1,B1} for the inner phrase.

The final noun introduces the category N_{e6} and adds the constraint *hat(e6)* to the CSP. The rule of application is used for the first time, closing the phrase, and producing the final category in (12e). Identifying e6 with e5 eliminates the box, B1, and constrains each candidate for e1 (= e2 = e3) to be in the remaining hat, H1. So R3 goes out. We are left with two singleton sets, thus ensuring the success of the two checks for definiteness. The conditions *unique(e5)* and *unique(e1)* are indeed enforced at this stage, because both the simple NP (corresponding to e5) and the complex NP (corresponding to e1) have been closed by the application to *hat*.

(12)	*Category*	*Constraints*	*Extensions*
a.	NP_{e1}/N_{e1}	*unique(e1)	
b.	$NP_{e1}/(N_{e2}\backslash N_{e2})$	rabbit(e2) e1 = e2	e1 : {R1,R2,R3}
c.	NP_{e1}/NP_{e4}	in(e3,e4) e2 = e3	e1 : {R2,R3} e4 : {H1,B1}
d.	NP_{e1}/N_{e5}	*unique(e5) e4 = e5	e1 : {R2,R3} e4 : {H1,B1}
e.	NP_{e1}	hat(e6) e5 = e6	e1 : {R2} e4 : {H1}

V CONCLUSION

A number of aspects of the present research have been put to one side in this paper. Nothing has been said about the treatment of sentences, in contrast to noun phrases, and little on the way in which certain structural ambiguities are resolved by reference to the context. Clearly there are phenomena for which the account sketched above is overly simple. Nonetheless, it is encouraging that a system of incremental evaluation can provide a solution to an instance of anaphora which poses a problem for more conventional approaches.

ACKNOWLEDGEMENTS

I wish to thank Mark Steedman and Henry Thompson for the supervision of this project. The work has also benefitted from the comments of: Gerry Altmann, Einar Jowsey, Ewan Klein, Colin Matheson, Chris Mellish, and Remo Pareschi. I am grateful to Jo Calder, Robert Dale, Jon Oberlander, and the two reviewers for reading and commenting on earlier versions of this paper. The work reported here was supported by SERC research quota award 83317765.

REFERENCES

Ades, A. and Steedman, M. J. (1982) On the Order of Words. *Linguistics and Philosophy*, **4**, 517-518.

Altmann, G. (1987) Modularity and Interaction in Sentence Processing. In Garfield, J. (ed.) *Modularity in Knowledge Representation and Natural Language Processing*. Cambridge, Mass.: Bradford/MIT Press.

Haddock, N. J., Klein, E. and Morrill, G. (eds.) (1987) Categorial Grammar, Unification Grammar, and Parsing. Technical Report No. EUCCS/WP-1, Centre for Cognitive Science, University of Edinburgh, 1987.

Hirst, G. (1983) Semantic Interpretation against Ambiguity. Technical Report No. CS-83-25, Dept. of Computer Science, Brown University, 1983.

Mackworth, A. K. (1977) Consistency in Networks of Relations. *Artificial Intelligence*, **8**, 99-118.

Mellish, C. S. (1985) *Computer Interpretation of Natural Language Descriptions*. Chichester: Ellis Horwood.

Pareschi, R. and Steedman, M. J. (1987) A Lazy Way to Chart-Parse with Extended Categorial Grammars. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, 1987.

Steedman, M. J. (1985) Dependency and Coordination in the Grammar of Dutch and English. *Language*, **61**, 523-568.

Steedman, M. J. (1987) Combinatory Grammars and Human Language Processing. In Garfield, J. (ed.) *Modularity in Knowledge Representation and Natural Language Processing*. Cambridge, Mass.: Bradford/MIT Press.

Winograd, T. (1972) *Understanding Natural Language*. New York: Academic Press.