A computer programme for sentence comprehension

Sascha W. Felix

This paper presents the outlines of a computer programme for the analysis of English sentences. The programme is written in BASIC and implemented on a Sharp MZ 80 B. Its theoretical orientation is Chomsky's theory of government and binding. The programme will thus distinguish between grammatical and ungrammatical sentences, give a structural description of input sentences and state reasons for ungrammaticalities. At present the programme handles successfully main and embedded clauses (long distance), wh-movement and NP-movement in passives.

Met hierdie artikel word 'n rekenaarprogram beskryf waarmee sinsontleding (Engels) gedoen kan word. Die program is in BASIC geskryf en word geïmplementeer op 'n Sharp MZ 80 B. Die teoretiese onderbou berus op Chomsky se teorie vir regering en binding. Gevolglik is die program in staat om te onderskei tussen grammatikale en ongrammatikale sinne, dit kan 'n strukturele beskrywing gee van gegewe sinne en redes aandui vir ongrammatikaliteite. Tans hanteer die program hoof- en bysinne (lang afstand), wh-verskuiwing en NP-verskuiwing in passiewe sinne.

1 Introduction

In recent years there has been a growing interest in the development of theoretical models for a human sentence parser (Fodor 1978; Frazier & Fodor 1978; Wanner & Maratsos 1978; Frazier et al. 1983; Johnson-Laird 1983). These models are theoretical in the sense that researchers have attempted to specify some general properties that must be minimally attributed to a parser which is to handle successfully sentences of a specific format. In general, these are sentences whose surface structure may lead a parser working linearly from left to right to choose an incorrect structural description without being able to discover the correct solution until at a much later parsing stage. Consider sentence (1) and (2):

(1) who did the teacher promise to reconsider the problem

(2) who did the teacher promise to write a book about

Since *promise* optionally takes an object, the parser may be led to assume that *who* is the object of *promise* by the time it arrives at the embedded sentence. This assumption is, of course, correct for (1), but incorrect for (2). In the latter case, however, the parser cannot discover its error until it has worked through to the end of the sentence.

Much current debate focuses on the question of what kind of linguistic theory a successful parser must be based on (Berwick & Weinberg 1983). Presently the two most likely candidates seem to be Gazdar's generalized phrase structure grammar (GPSG), (Gazdar 1982) and Chomsky's theory of government and binding (GB), (Chomsky 1981).

This paper is a first report on a project currently pursued at the University of Passau in which we attempt to develop a computer programme that reflects some of the major theoretical considerations of the parsing literature. In order to see at the same time whether or not such a goal can be achieved with fairly simple and unsophisticated equipment, we use a personal computer Sharp

MZ 80B and BASIC as the programming language (BASIC = Beginner's All-Purpose Symbolic Instruction Code).

Although it is obvious that the present programme is far from being an even remotely psychological model of human sentence parsing, we have imposed a number of basic requirements on the programme, which, in a sense, reflect typical properties of human sentence comprehension. Following this guideline we eventually hope to gain insight into human parsing by studying the formal properties of a successful computer programme.

The primary requirement is, of course, that the computer system should be able to assign the correct structural description to the sentence to be parsed. However, the system should also be able to handle ungrammatical sentences and identify them as such. That is, the system should provide grammaticality judgments on sentences. As a final requirement, the system should identify different kinds of ungrammaticality. Consider (3)–(5):

(3) *loves Mary old the man

(4) *who does Mary love the man

(5) *who does Mary love John and

Even though (3)–(5) are all ungrammatical, speakers of English have fairly clear intuitions about the fact that these sentences are ungrammatical for different reasons. In a similar manner, we expect the computer system not only to identify ungrammatical sentences, but also to specify the kind of ungrammaticality.

The linguistic theory which the system operates on is Chomsky's theory of government and binding (1981). The crucial feature of this theory is that it contains not only rules of grammar but also conditions for well-formedness, that is, constraints on the output of the rules. Consequently, one part of the computer programme will analyse the input sentence according to these rules, that is assign a structural description to the sentence, and another part will check whether or not the structural description meets the conditions for well formed syntax. It is thus obvious that an input sentence may fall through at two different levels. Either the system will not be able to assign a structural description to the input sentence, which would be the case, for example, in (3), or the structural description will fail to meet any one of a certain number of output constraints which would happen for sentences such as (4) and (5). Depending on where the sentence falls through, the system will identify different kinds of ungrammaticality. Further details will be specified in section 2.

It is self-evident that in developing a computer programme for linguistic analysis one has to decide what a programme, as yet incomplete, should be able to do and what appears to be dispensable at the present stage. So something needs to be said about the possibilities and limitations of the programme as it presently stands.

Some limitations are purely system-internal. For example, no input sentence may be longer than 53 symbols (including spaces), which simply reflects the fact that in the system we are using, alphanumeric variables are limited to that length. For longer sentences one would need more than one variable. In principle, this is no problem, but it would merely make our problem more complex at a level which does not seem very interesting.

Other limitations reflect deliberate decisions of preference on our part. First, the system ignores morphology. It does not distinguish singular from plural, nor past tense from present tense. Since we are primarily interested in syntax, in particular in the types of structures that have motivated recent research in parsing mechanisms, this limitation does not affect our principal goal.

At present the lexicon comprises sixty two items, so the system will accept relatively few sentences. To those familiar with the linguistic literature it will come as no surprise that most sentences are about John loving Mary or about someone saying or believing so. Expanding the lexicon is, of course, a matter of spending a few more hours on routine work rather than on creative work.

Finally, the system does not accept more than one auxiliary or adverbials or prepositional phrases for which the verb is not subcategorized. This latter fact is obviously a more serious limitation which will have to be remedied as soon as the programme accomplishes the essentials.

Apart from these limitations the system accepts sentences of any complexity. In particular, it can handle any type of constituent co-ordination and any number of sentence embeddings, except relative clauses. It furthermore accepts passives as instances of NP-movement and it handles wh-movement including long distance extractions. Of course, these are all fairly modest accomplishments and it is clear that there is more the system can't do than what it can do. However, the way the system does what it can do provides interesting insights into how a parser has to be constructed.

2 Programme structure

The programme consists of three essentially independent parts:

(1) *preparation routine*

(2) *parsing routine*

(3) *well-formedness routine*

The motivation of the first part is purely system-internal, since the computer accepts the input sentence merely as an alphanumeric variable consisting of a certain number of symbols. Before the actual analysis can start, it is thus necessary to segment the input sentence into words and to identify the word class of each lexical item. This is accomplished in the preparational routine. The second part contains the parsing routine. Its input is a sequence of alphanumeric variables, each variable representing one lexical item associated with the word class it belongs to. The output of the parsing routine is conventional labelled bracketing. If a sentence has been successfully parsed – which might not be the case, as we will see in section 2.2 –, the labelled bracketing will be transferred as input to the third part. The well-formedness routine checks the sentence structure for possible violations of principles of universal grammar. If no violations are found the sentence will be qualified as "grammatical", otherwise as "ungrammatical" with indications of which principle(s) is/are violated. I will discuss these three parts of the programme in turn.

2.1 The preparation routine

After displaying the main menu, the system will first read the lexicon which at present contains sixty two items and store it in a specific memory area for immediate access. Each lexical item is associated with a set of four variables. The first specifies the word class; the second various information data as, for example "operator" in the case of wh-words. The third variable contains the lexical item's subcategorization frame (if any). The fourth variable is currently empty (= O). The system will now ask for the sentence to be analysed and this is entered in conventional spelling. After the input has been completed, the system will segment the sentence into words by simply using spaces as cues. That is, any sequence of letters surrounded by spaces is taken to be a word. Subsequently the list of words identified is displayed on the screen accompanied by the option of returning to the input routine in case of error.

The next step is to identify each word's lexical category and, again, to display the result on the screen. This is done essentially by running through the lexicon memory and by storing the contents of the first variable as soon as the lexical item has been found. If the sentence contains a word that is not in the lexicon, the display will be "no lexical entry for X". In this case it is possible to return to the input routine for a new sentence.

This completes the preparation routine of the programme. The final display of this part is a concatenated sequence of lexical category symbols, for "Mary loves a nice boy" it will be N+V+Det+A+N.

Internally the system will work with this sequence of category symbols rather than with the sequence of actual lexical items. It returns to the lexical item only if decisions have to be taken that depend on subcategorization information. This is at present only the case with verbs. That is, if the system encounters the category symbol V, it will look which actual lexical item V stands for, and will then decide, on the basis of that particular item's subcategorization frame, on the next step. It is clear that this routine can and must be extended to other lexical categories, for example nouns, ("the claim that S").

2.2 The parsing routine

The parsing part of the programme is essentially a top-to-bottom and left-to-right routine, working in the way of an ATN (=Augmented Transitional Network) model. That is, the system runs through a prescribed number of

networks and attaches the network to the tree-structure once the relevant parses are completed.

At the present stage of our work the system is heavily biased as to what a possible phrase structure tree may look like. It will therefore ignore anything that does not fit its biases, which is an admittedly serious limitation, since any optional constituents that a sentence may contain, in particular adverbials and non-subcategorised prepositional phrases, are simply thrown out during that parse. Under a different perspective we may say that the system is limited to analysing only those elements of the predicate that are dominated by VP or rather V, ignoring all others. However, it should not be difficult to extend the programme to accept also VP-constituents that are optional. In principle, the system should merely close the V after analysing the obligatory predicate constituents, attaching all the rest together with V to V.

The system will first enter the S/COMP network for the analysis of the topmost sentence. If during the subsequent parses the system encounters a new sentence boundary it will return to the S/COMP network starting over again. In this way an unlimited number of embedded sentences can be analysed in principle, although for reasons of memory capacity the actual number is at present limited to two.

The system knows that the initial position of any S is COMP. Currently the system can only identify *that* or wh-words as potential elements of COMP (topicalisations can therefore not be handled). It will thus check if the sentence-initial item is one of these elements and, if so, attach it to COMP. If the system finds *that* in initial position of the topmost S, the sentence will be labelled as "ungrammatical" during the well-formedness routine which will be discussed in detail in section 2.3. For subject sentences ("that John loves Mary surprises me") there will be a restructuring process, so by the time COMP violations are checked, *that* will no longer be in the topmost S. If the system does not find any element that qualifies for COMP, it will assign "e" (empty) to that node.

Since the system knows that COMP is followed by the subject NP it will next enter the NP network. The computer will scan through the subsequent lexical categories until it finds a noun, assuming that that noun is the head of NP. It will further assume that all elements between COMP and the noun are members of the NP. If the noun is the only member, it will immediately attach it to NP.

The system will expect only Det, A and/or Adv to the left of the noun and will analyse the NP accordingly. Co-ordinations are also adequately handled within this routine. If the system encounters a co-ordinator (*and/or*), it will know that the element following the co-ordinator has to be the same as the one preceding it. If it does not find an identical element (e.g. *the nice and man), it will construct one to which it assigns an empty category (EC). In this case the sentence will be ruled out as ungrammatical in the well-formedness routine. Co-ordinated determiners ("the and a man") are ruled out independently.

After this system has worked through all the relevant items it will check whether or not the first noun is followed by a co-ordinator, in which case a further NP is expected and the routine starts over again. Otherwise the system will leave the NP network.

Let us assume that the system does not find a noun between COMP and the first verbal element as in *who loves Mary* or ... *loves Mary old the man*. In this case it will assume a phonetically unrealised NP to which it assigns an EC.

The system now enters the VP network looking for the first verb category. It will then check if there is an AUX to the left of the verb. If there is not, it will again assign an EC to AUX. Once the first verb has been found the system will look up its subcategorization frame. Suppose the verb is transitive. In this case the system will temporarily return to the NP network. If it doesn't find an NP as in *who does John love*, it will again assign an EC to that NP. In general the system will assign an EC to any syntactic category that it expects on the basis of subcategorisations, but doesn't find. Suppose the verb is intransitive. In this case the system closes the VP and will therefore simply ignore any NP that may nevertheless appear.

This procedure has some interesting consequences because subcategorisation violations will be recognised in different ways. Consider (6)–(8):

(6) *who does John see Mary

(7) *John sees

(8) *John runs Mary

The system will correctly reject all three sentences as ungrammatical, but in each case for a different reason. (6) and (7) will pass the parsing routine and be rejected in the well-formedness part in ways to be explained in section 2.3. (8) will be thrown out within the final part of the parsing routine to which I will shortly return.

If the verb is subcategorized for S, the system will return to the S/COMP network and start the routine again in obvious ways. If the verb is subcategorised for more than one category, or for different categories, the system will try out all the various possibilities and look for the respective categories.

At the end of each parsing step as well as the entire procedure the labelled bracketing which the parser has constructed will be displayed on the screen. Crucially, the labelled bracketing may contain more or less material than the original input sentence. It will contain more material if empty categories have been assigned for phonetically unrealised constituents, and less material, if the parser encounters material which it cannot integrate into the tree and therefore ignores. Consider again sentence (8). After the parser has correctly analysed *John runs,* the V will be closed, since run is an intransitive verb. The following NP *Mary* is thus ignored because it does not fit into the tree. For an input sentence such as (8) the parser will simply yield the output (9):

(9) [[John] [[e] [runs]]
S NP VP AUX V

In the final step of the parsing routine the system will compare the lexical material in the output, that is, *John runs*, with the input sentence, that is *John runs Mary*. If there is a mismatch between the two, that is, if the input sentence contains lexical material that does not reappear in the parsing result, the sentence will be classified as ungrammatical with the additional indication that it is "unparsable". Unparsable in this sense implies that the parser found material in the input sentence which – in plain language – it did not know what to do with. This will happen in case of certain types of subcategorisation violations and with "mixed-up" word orders.

2.3 Well-formedness routine

If a sentence has successfully passed the parsing routine, its labelled bracketing is transferred as input to the well-formedness routine. The output of this part is a grammaticality statement (grammatical versus ungrammatical). If a sentence is found to be ungrammatical, the system indicates which principle of Universal Grammar is violated.

It seems obvious that this part of the programme will eventually turn out to be the most complex routine. At the present stage of our work there are only three areas in which the system is able to detect ungrammaticalities.

First, the topmost sentence is checked for the presence of a complementiser. If it finds, for example *that*, the sentence is classified as ungrammatical. Second, the system checks for empty NP's in co-ordination, in which case the sentence is again labelled as ungrammatical. This subroutine has eventually to be integrated into a check-up for subjacency violation which we have not completed yet. The major part of the well-formedness routine concerns wh-movement, that is the appropriate binding of variables. The system first determines whether or not there is an operator (= wh-word) in COMP. If there is one, the system will scan through the tree structure for empty NPs. If it does not find one, the operator does not bind a variable, thus violating the binding theory and the bijection principle (Chomsky 1982). Under this subroutine a sentence such as (6) will be recognised as being ungrammatical. If the labelled bracketing of a sentence such as (7) contains an empty NP (= the object), but lacks an operator in COMP, this EC will be unbound, violating Principles A and B of the binding theory (Chomsky 1982). Consequently this sentence will be recognised as ungrammatical, too, because of binding violations. More generally speaking, the system checks if each operator binds a variable and if each variable is bound by an operator.

This completes the description of the major features of the programme as it currently stands. We are at this moment working on a subroutine for NP-movement, that is binding of anaphors in accordance with Principle A. I will therefore discuss a few problems faced in this area, and the way in which we plan to solve them.

The system has a few problems in handling NP-movement in simple passives and raising-verb-constructions such as (10) and (11):

(10) John was killed *e*

(11) John seems *e* to love Mary

The system must only know that ECs in the context of passive and raising constructions reflect NP-movement rather than wh-movement and can then check proper binding in ways similar to the previous cases. The crucial problem arises if a sentence contains more than one EC, one of which is a variable, the other an anaphor as in (12):

(12) who does John seem e to love *e*

The system has to know that the first EC is the trace of *John*, while the second is a variable bound by *who*, and not the other way round.

In the case of (12) a simple solution suggests itself. If the number of ECs matches the number of possible binders, the sentence will be grammatical; that is in (12) we have two ECs and two binders (*John* and *who*). Of course, this array of facts is purely accidental as (13) shows, where we have again two ECs but only one lexical binder (*John*):

(13) John seems e to have been killed *e*

The correct solution is, of course, that the system *must* be able to distinguish between anaphors and variables. Crucially, this has to be accomplished by means independent of the presence versus absence of lexical binders.

We are presently working on a solution that relies essentially on Case Theory. Anaphors differ from variables in that the former don't have case, whereas the latter do. This distinction can be determined by the system on the basis of the available labelled bracketing. Thus in (12) the subject-EC does not have Case since the verb is non-tensed, whereas the object-EC receives Case from *love*. The system therefore knows that the first EC must be an anaphor while the second is a variable. Hence it can look for the proper binder in each case. In (13) neither EC receives Case and therefore must be an anaphor.

If this solution turns out to be correct, it will have some interesting consequences, namely that Case Theory must apply before Binding Theory. In other words, different principles of Universal Grammar may have a different status, an idea which may be worth while exploring.

3 Conclusion

In this paper we have presented the outlines of a computer programme for the analysis of English sentences. The basic structure of this programme derives from recent considerations in linguistic theory. In particular, the programme takes grammar to be an essentially independent module and distinguishes furthermore between parsing as structure-building governed by rules of grammar and checking well-formedness constraints on output structures.
Although the programme is still in the early stages of development, the general approach appears to be quite promising.

Bibliography

BERWICK, R. & WEINBERG, A. 1983. The role of grammars in models of language use. *Cognition* 13.

CHOMSKY, N. 1981. *Lectures on government and binding*. Dordrecht: Floris.

CHOMSKY, N.b 1982. *Some concepts and consequences of the theory of government and binding*. Cambridge, MA: MIT press.

FODOR, J. 1978. Parsing strategies and constraints on transformations. *Linguistic Inquiry* 9.

FRAZIER, L., CLIFTON, C. & RANDALL, J. 1983. Filling gaps: decision principles in sentence comprehension. *Cognition* 13.

FRAZIER, L. & FODOR, J. 1978. "The sausage machine": a new two-stage parsing model. *Cognition* 6.

GAZDAR, G. 1982. Phrase structure grammar. JACOBSON, P. & PULLUM, G. (eds), *The nature of syntactic representation*. Dordrecht: Floris.

HALLE, M., BRESNAN, J. & MILLER, G. (eds) 1978. *Linguistic theory and psychological reality*. Cambridge, MA: MIT press.

JACOBSON, P. & PULLUM, G. (eds) 1982. *The nature of syntactic representation*. Dordrecht: Floris.

JOHNSON-LAIRD, P.N. 1983. *Mental models*. Cambridge: Cambridge University Press.

WANNER, E. & MARATSOS, M 1978. An ATN approach to comprehension. In: HALLE, M., BRESNAN, J. & MILLER, G. (eds), *Linguistic theory and psychological reality*. Cambridge, MA: MIT Press.