SOME REMARKS ON THE IMPLICATIONS OF THE DOUBTFUL GAP PROBLEM FOR HUMAN SENTENCE PROCESSING[1]

Hans van de Koot

Abstract

In this paper I argue that, contrary to popular opinion, the so-called doubtful gap problem is not beyond the abilities of a strictly deterministic parser such as Marcus' (1980) Parsifal. I develop a chain algorithm for this parser that is nonstandard in that it divorces the process of trace indexing from that of trace creation. This algorithm will account for the data, on condition that sentences with multiple doubtful gaps (i.e. with globally ambiguous chains) do not exist. It is also shown that a set of apparent counterexamples can be reanalyzed as parasitic gap structures.

1 Introduction

In this paper I will be concerned with a special type of parsing problem that is often referred to as the doubtful gap problem. One of the many manifestations of this phenomenon is illustrated in (1) below:

(1) (i) who$_i$ did you expect [$_{IP}$t$_i$ to make a potholder]
(ii) who$_i$ did you$_k$ expect [$_{CP}$[$_{IP}$PRO$_k$ to make a potholder for t$_i$]]

Whatever the exact properties of the human parser one assumes, upon encountering the verb *expect* it has to decide whether to create a CP-complement with a PRO subject or whether the complement is in fact an IP, the specifier of which is the foot of the chain headed by *who*. It should be observed that this decision is *locally* ambiguous: if the parser had access to the relevant information in the part of the sentence to the right of its current locus of attention, then it could resolve the ambiguity. Unfortunately, problems of this kind are potentially very difficult because the disambiguating material can be arbitrarily far away from the point of local ambiguity.

As we will see shortly, the apparent lack of boundedness makes the doubtful gap problem a serious challenge for researchers who have claimed that human sentence processing is strictly deterministic (in the sense that the parser does not simulate nondeterminism by means of backtracking or parallel processing[2], e.g. Marcus 1980 and Berwick & Weinberg 1984). It would in fact

---

[1]This paper is an extensively reworked version of a talk that I gave to the Annual Meeting of the Dutch Linguistic Society, which was held at the University of Leiden on the 21st of January 1989.

[2]In the case of backtracking, the program is constructed in such a way that it orders its hypotheses about the input and then tests each hypothesis against the input string in turn. The program accepts the input if one or more of its hypotheses lead

seem that the deterministic parsers proposed by these authors cannot deal with the full range of doubtful gaps that we are about to consider. This has led many researchers to the conclusion that the human sentence processor must be a parser of the backtracking variety or one that can carry along more than one parse tree in parallel. The aim of the present paper is to show that that conclusion does not necessarily follow.

In section 2, I first introduce some of the basic properties of the Marcus parser. This is followed in section 3 by a survey the range of problems introduced by doubtful gaps. It will become clear that the strictly deterministic parsers proposed by Marcus (op. cit.) and Berwick and Weinberg (op. cit.) dealt with the doubtful gap problem in a much simplified form. These authors only concerned themselves with 'easy' cases of *wh*-movement from subject or object position, and even for these simplest cases they ignored doubtful gaps that are disambiguated by material to the right rather than by material to the left of the gap. My survey is based on part of Fodor's (1978) article 'Parsing Strategies and Constraints on Transformations', which contains many important data and a good survey of parsing strategies that do not properly account for them.

With this background we take a closer look at strictly deterministic parsers in section 4. Can they be made to account for the facts? I propose an alternative chain algorithm that will do the job, provided that sentences with multiple doubtful gaps do not exist. The algorithm is based on a suggestion by Rizzi (1988) that referential indices must be licensed by a θ-role. I translate Rizzi's idea into a condition on trace indexing, which, in conjunction with a quite straightforward assumption about the visibility of elements at LF, yields the desired consequences.

As is pointed out in Barton, Berwick and Ristad (1987), powerful problem solving techniques (like backtracking) implicitly assume that 'the natural problem has no *special structure* that would support more specialized processing methods - despite all the principles, representational details, nooks, and crannies of grammatical theory'. We will see that the doubtful gap problem has a special structure that will yield to something less computationally expensive than backtracking.

Finally, we compare our solution to a modification of the Marcus parser proposed by Baart and Raaijmakers (1988), which, like ours, is intended as an antidote to the doubtful gap problem, and show that Baart & Raaijmakers' algorithm creates as many problems as it solves. This will leads us into a discussion of some data that appear to involve multiple doubtful gaps and therefore pose a threat to our analysis. I argue that these 'multiple doubtful gaps' are really parasitic gap constructions.

I conclude that the proposed algorithm allows a strictly deterministic solution to the problems posed by doubtful gaps, that it can be readily incorporated into the Marcus parser, and that it avoids the unattractive efficiency consequences of implementing backtracking search or full-scale parallelism.

---

to a success state. Alternatively, the program could pursue all alternative hypotheses in parallel. Here, too, we say that the program accepts the input if one or more alternative pathways lead to a success state. The Marcus parser is deterministic in the nontrivial sense that it uses neither backtracking nor parallel processing to simulate nondeterministic operation. Marcus uses the notion "strict determinism" to refer to this much narrower concept and we will follow this practice here.

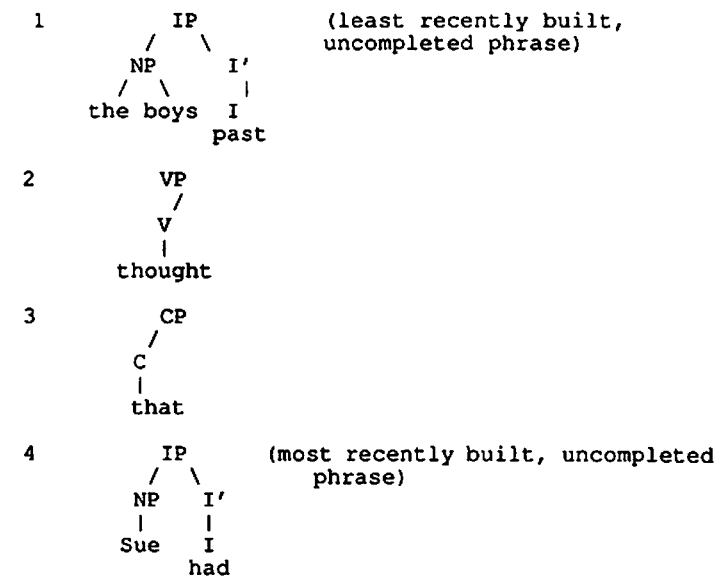
2 The Marcus parser

The Marcus parser has been developed on the basis of the Determinism Hypothesis: 'the syntax of any natural language can be parsed by a mechanism which operates strictly deterministically in that it does not simulate a nondeterministic machine' (Marcus 1980: p.2). Every action of a strictly deterministic machine is completely determined by its input and its internal state. This means that it cannot make 'guesses' about the proper sequence of action for a given input. Thus, such a machine is not allowed to compute two or more sequences of actions in parallel (parallel processing) or to withdraw a sequence of actions and make a new 'guess' in case it fails to accept a given input (backtracking). The Marcus parser operates strictly deterministically in that (i) it builds only one syntactic tree representation of a sentence, (ii) it does not carry along in parallel a representation of other possible parses, and (iii) it cannot withdraw previous decisions about the structure of sentences. These three properties help to guarantee that the parser uses only a limited number of memory cells. This restriction on memory capacity is essential: the Determinism Hypothesis would be greatly weakened if the parser were allowed to circumvent guesses by means of a very large memory capacity.

Marcus shows that a deterministic machine must have the following three properties: (i) it must be at least partially data driven, i.e. its actions must be partly determined by the input, (ii) it must be able to reflect expectations on the basis of the partial structures built up during the parse, and (iii) it must have some sort of look-ahead facility. The Marcus parser uses two very simple data structures from which it derives these properties. First, it has access to part of the parse tree it has built so far, the Active Node Stack. This stack of uncompleted nodes is the parser's 'left context'. The stack records the parser's expectations about the input by associating with each node in the stack a set of rules that are needed to build that node. Such a set of rules is called *active* when the node with which they are stored becomes the current node (i.e. the one currently being built). The parser is also equipped with a look-ahead buffer consisting of only three memory cells. The buffer is the parser's 'right context' and represents its data driven mode of operation. Marcus claims that these information facilities suffice to ensure that 'all sentences which people can parse without conscious difficulty can be parsed strictly deterministically' (Marcus 1980: p.6).

The two data structures are manipulated by the parser's grammar, which operates by attaching constituents from the buffer to the constituent at the bottom of the stack (we will think of the stack as growing *downward*) until that constituent is complete, at which time it is popped from the stack and inserted in the first cell of the buffer. If the constituents in the buffer provide clear evidence that a constituent of a given type should be initiated, a new node of that type can be created and pushed onto the stack. It is not particularly relevant for what follows that one has detailed knowledge of the parser's grammar rules, so I will not say more about them. However, it would be helpful if the reader had a rough understanding of what a stack representation in the Marcus parser looks like, I present an example below (from Berwick & Weinberg 1984: p.151). By the time the parser is reading *kissed* in *the boys thought that Sue had kissed them*, it has already built up a parsed representation of the sentence as in (2):

(2) Active Node Stack (4 deep)



When the current node in the stack (i.e. the most recently built node) is complete, it is popped from the stack and dropped in the first cell of the parser's buffer for subsequent attachment to the new current node. When the parse is ready, the complete structure for the input sentence will be in the first cell of the buffer and the stack will be empty.

Berwick and Weinberg's investigation is based on a slightly modified Marcus parser which realizes a version of the Government Binding Theory. Even though there is no one-to-one relation between rules and operations of the grammar and rules and operations of the parser, the modified Marcus parser makes direct use of the grammar in that it makes crucial reference to properties of lexical items, in accordance with the Projection Principle (Chomsky,1981). More generally, the parser embeds the key principles of Government Binding Theory fairly directly.

The main problem for a strictly deterministic parser is local ambiguity. The parser can construct just one syntactic tree representation for every sentence and it cannot undo previous actions. This means, in effect, that the parser must resolve any ambiguity it comes across before it can continue the parse. Consider the following data:

(3) (i) Who$_i$ did John kiss t_i

(ii) What$_i$ did John say t_i that Frank believed t_i that Sue said . . . t_i that Bill would like to eat t_i

(iii) Did John say that Frank believed that Sue said that Bill would like to eat

Structure (3i), in particular the insertion of the postverbal trace, is unproblematic for the parser, as it has direct access to the properties of lexical items. Hence, recognition of the verb *kiss* entails retrieval of this item's subcategorization frame as well. As *kiss* is a transitive verb, the parser will expect to see a NP next. If there is no phonological NP, then the parser will attach an empty NP (a trace) as the object of *kiss*. Note that strictly local information (i.e. government) precludes insertion of PRO. Now consider (3ii). *Eat* may be either transitive or intransitive. Given the assumption of strict determinism, the parser must decide what to do correctly before it can proceed to the next item in the input stream. Clearly, the ambiguity can only be resolved by consulting the left context to see if there is an appropriate antecedent. If an appropriate antecedent is found, the parser will assume that *eat* is transitive and insert a trace. If, on the other hand, the left context does not contain an appropriate antecedent, the parser will do nothing and move on to the next item in the input stream.
As can be seen in (3ii), the site for trace insertion can be arbitrarily far away from its phonological antecedent. This means that the relevant parsing rule would have to refer to a potentially unbounded left context (assuming now that there are no intermediate traces). Such a rule could be finitely stated in terms of an essential variable, but this option is not available to the Marcus parser. This is because the parser's stack is manipulated by a finite control table that only uses actual grammar symbols. It follows that a strictly deterministic parser can only detect an antecedent in a literally finite left context. Hence, the parser must have a trace insertion procedure for intermediate traces.

It should be evident from the foregoing that the property of strict determinism imposes *some* locality constraint on trace insertion. Both Marcus (1980) and Berwick & Weinberg (1984) go in fact one step further in claiming that a strictly deterministic parser *explains* the existence of the Subjacency Condition on overt syntactic movement:

(4) Subjacency
No rule can involve X and Y in the configuration:
$[...X...[_{\alpha}...[_{\beta}...Y...]...]...X...]$
where α and β are bounding nodes.

For critical discussions of this claim I refer the reader to Fodor (1985) and Van de Koot (1985, 1987, to appear). In what follows I will assume the analysis put forward in Van de Koot (to appear) that a suitably restricted version of the Marcus parser imposes a constraint on syntactic movement that is much stronger than Subjacency. What the proposed restriction amounts to is that, in addition to the buffer cells, only the current node in the stack should be accessible to parsing rules[3]. It follows that the parser will impose a locality constraint on traces left by syntactic movement that is similar to antecedent governed. On this view a simple case of *wh*-movement like (5i) will be assigned

[3]In Parsifal and in Berwick & Weinberg's modified version of that parser both the current node and the so-called *current cyclic* (the first cyclic node in the stack) were accessible to parsing rules. This rendered the stack non-standard, to say the least (as was acknowledged by Marcus). My "restriction" is therefore not really a restriction: it just amounts to stating the obvious, viz. that a stack is a last-in-first-out memory device.

the structure (5ii):

(5) (i) who did John kiss
(ii) who$_i$ did [$_{IP}$ t_i [$_{IP}$ John [$_{VP}$ t_i [$_{VP}$ kiss t_i]]]]

In other words, the only way in which the parser can encode the presence of an antecedent is by adjoining a trace to every maximal projection c-commanded by the antecedent, until a gap is detected. This proposal would seem to run into rather obvious problems with respect to A-chains. Consider a standard case of NP-movement:

(6) John was believed [$_{IP}$t' to have been [$_{VP}$kidnapped t]]

NP-movement is not assumed to leave a trace in every maximal projection that intervenes between the head and the foot of the chain. In terms of the proposed parser this has the undesirable consequence that neither of the two traces indicated in (6) has an accessible antecedent at the point where it should be created.

In order to obviate these problems, I assumed (i) that the SPEC of VP and the SPEC of AP are A-positions, and (ii) that adjoined positions are A'-positions, as is standard. The first of these assumptions is based on the idea that the position of subjects at D-structure is specVP universally, as has been proposed by Kuroda (1986)[4]. Kuroda points out that the conceptual simplification that results from including the expansion of C and I in the general X'-schema is considerably weakened by the fact that the category V conforms only defectively to the schema of the extended X'-theory. His paper investigates the consequences of the conceptually simplest assumption, viz. that specVP is the D-structure subject position, with some elegant results.

Given the above assumptions about specVP, the parser will be able to construct a proper chain for NP-movement cases, because in cases where raising or passive is possible specVP/specAP is a θ-bar position:

(7) (i) [$_{IP}$John was [$_{VP}$t killed t]]
(ii) [$_{IP}$John [$_{VP}$t seems [$_{IP}$t to [$_{VP}$t win]]]
(iii) [$_{IP}$John [$_{VP}$t is [$_{AP}$t likely [$_{IP}$t to [$_{VP}$t die]]]]

Second, super-raising and super-passive are excluded for two independent reasons:

(8) (i) * [$_{IP}$John [$_{VP}$$t_i$ seems [$_{CP}$(t_i) that [$_{IP}$it [$_{VP}$$t_i$ is [$_{AP}$$t_i$ certain [$_{IP}$$t_i$ to [$_{VP}$$t_i$ win]]]]]]]]
(ii) * [$_{IP}$John [$_{VP}$$t_i$ seems [$_{CP}$(t_i) that [$_{IP}$it [$_{VP}$$t_i$ was [$_{AP}$$t_i$ killed t_i]]]]]]

First, in both (8i,ii) a C-projection intervenes, which does not have an A-position. Hence, chain formation is blocked and both sentences are ruled out. Second, in both (8i,ii) one of the intermediate positions (specIP) is filled,

[4]Koopman and Sportiche (1985) propose an analysis that is rather similar to Kuroda's, except that in their view the D-structure subject is generated in a VP-adjoined position, i.e. it is the subject of a small clause.

effectively blocking chain formation. I.e., we assume that chain links are coindexed at the moment of their creation. Hence, a dummy element like *it* cannot be coindexed with a governing trace, as it is part of the parser's input and not a created chain link[5].

Before we move on to consider the full range of doubtful gap problems, let quickly summarize the main points sofar:

(i) Deterministic parsers have very limited access to their left and right context;

(ii) Deterministic gap detection relies on a chain algorithm that creates intermediate traces to guarantee the local availability of antecedents;

(iii) If a deterministic parser detects a potential gap and its current node contains a proper antecedent, then it inserts a trace. It is this behaviour that allows it to handle locally ambiguous cases like (3ii,3iii) above.

3 The doubtful gap problem: Fodor (1978)

In her article 'Parsing Strategies and Constraints on Transformations', Fodor (op. cit.) shows that gap filling introduces parsing difficulties that have no parallel in the grammar. As an example, consider the data below[6]:

(9) (i) who$_i$ did you expect [$_{IP}$t$_i$ to make a potholder]
(ii) who$_i$ did you$_k$ expect [$_{CP}$[$_{IP}$PRO$_k$ to make a potholder for t$_i$]]

When the parser is analyzing expect, it must decide whether to create an IP-complement with a trace in its specifier or a CP-complement with a PRO subject. But in order to do so, it must have access to material that it has not seen yet, i.e. material in its *right* context. Clearly, gap filling is not as easy as it might have seemed from such data as in (3). As another example, consider (10). Here, too, the parser must have access to material to the right of the doubtful gap in order to resolve the local ambiguity induced by this gap:

(10) what$_i$ do you want Mother to sing (t$_i$) to Mary about t$_i$

Fodor discusses three models of gap finding that use different strategies to deal with doubtful gaps: the last-resort model of gap finding, the first-resort model of gap finding, and the lexical-expectation model of gap finding. We will discuss each of these in turn.

3.1 The last-resort model of gap finding. A parser with a last-resort strategy for gap detection will only assume a gap in the input if all other structural hypotheses about the locally ambiguous chunk have failed. This means that it will always detect doubtless gaps on its first pass through the sentence, but that it will overlook all doubtful gaps. Obviously, if the doubtful gap was a

[5]It is, we bar accidental coindexing.

[6]All data in this section are from Fodor (1978), unless indicated otherwise. Fodor discusses these data in terms of Aspects style rules. I will discuss her data in terms of current grammatical notions.

false one, the parser outputs a correct derivation and even may have saved processing effort. On the other hand, if the doubtful gap was a true one, the analysis will halt at some point and will have to be revised. This model therefore predicts that (a) sentences in which the true gap is a doubtful gap should always be harder to parse than sentences in which the true gap is a doubtless gap and furthermore that (b) sentences with a false doubtful gap should be just as easy to parse as sentences with no doubtful gap at all. Unfortunately, these predictions seem to be false. The pair of sentences (11) is a counterexample to prediction (a):

(11) (i) which book$_i$ did the teacher read (t_i) to the children
(ii) which picture$_i$ did the teacher show t_i to the children

Intuitively, sentence (11i) is just as easy to parse as (11ii). However, and Fodor acknowledges this point, one might argue that our intuitive judgements of processing complexity are just not sophisticated enough to reveal a difference. She counters this view of the matter by arguing that it does no justice to our perception of the quite comparable sentences in (12):

(12) (i) which book$_i$ did the teacher read (t_i) to the children from t_i
(ii) which student$_i$ did the teacher go to the concert with t_i

Fodor argues that 'there is a rather clear intuition that [(12i)] is harder to process than [(12ii)]'.

These sentences are also counterexamples to prediction (b), since the relative difficulty of (12i) must apparently be attributed to the false doubtful gap:

> 'This gap is encountered before the true gap, and apparently 'decoys' the parser; it seems to be taken at least momentarily as the true gap, giving the analysis *The teacher did read which book...*, which must subsequently be corrected when the true gap is found after *from*. This explanation is supported by the fact that [(12i)] is also more difficult than [(11i)]. [(12i)] and [(11i)] start out identically, and it seems that in both the doubtful gap is noticed and hypothesized to be the true gap. For [(11i)] this hypothesis is correct and the analysis proceeds smoothly; but for [(12i)] this hypothesis turns out to be wrong and the analysis is disrupted.'

She concludes that, since all doubtful gaps she has discussed are detected as they are encountered, the last-resort model must be false.

3.2 **The first-resort model of gap finding.** The first-resort strategy is exemplified in the original Marcus parser and in both of the modified Marcus parser that we have mentioned (Berwick & Weinberg 1984 and Van de Koot (to appear)): these parsers assume a gap if the left context of the parse and local evidence is compatible with that assumption. By compatibility with local evidence I mean that the parser should not drop a trace for an antecedent of category X if the hypothetical gap position is immediately followed by a phrase of category X. The first-resort model predicts that (a) sentences with a true

doubtful gap are as easy to parse as sentences with a true doubtless gap and that (b) sentences with a false doubtful gap should be harder to parse than sentences containing no doubtful gap at all. Even though these predictions are compatible with the sentences in (11) and (12), other examples show that they are incorrect:

(13) (i) which student$_i$ did the teacher walk (t_i) to the cafeteria
(ii) which student$_i$ did the teacher walk (t_i) to the cafeteria with t_i

Fodor suggests that (13ii) is easier to parse than (13i) and that this is explained on the assumption that the parser overlooks the doubtful gaps in both sentences, which leads to error in (13i) but facilitates the analysis of (13ii). Therefore, the first-resort model must be false as well. If we now compare (13i) with (11ii), then we see that prediction (a) of the first resort model is disconfirmed by the fact that the doubtful gap in (13i) is harder to process than the doubtless gap in (11ii). Similarly, prediction (b) is disconfirmed by comparing (13ii) and (12ii): (13ii) is not harder to process than (12ii), which suggests that its doubtful gap does not decoy the parser.

3.3 The lexical-expectation model of gap finding. Fodor concludes that, although neither the last-resort nor the first-resort strategy seems to be completely correct, the data nevertheless suggest that the human sentence processor 'behaves like a first-resort device for some gaps, and like a last resort device for others'. She then points out that Wanner and colleagues (in a series of papers) have developed a model that will account for all the data she discusses ('though they themselves make no mention of it'). The proposal of Wanner and colleagues is to associate each lexical item with a ranking of preferred complements. In particular, a verb like *read* will be associated with the ranking in (14), while a verb like *walk* will be associate with the ranking in (15) (where A and B stand for alternatives such as DET, N, or Pronoun, and Ø stands for the empty set):

(14)

READ:

$$\begin{bmatrix} \text{NP} = \begin{vmatrix} \text{A} \\ \text{GAP} \\ \text{B} \end{vmatrix} \\ \emptyset \end{bmatrix}$$

(15)

WALK:

$$\begin{bmatrix} \emptyset \\ \text{NP} = \begin{vmatrix} \text{A} \\ \text{GAP} \\ \text{B} \end{vmatrix} \end{bmatrix}$$

Unambiguously transitive or intransitive verbs represent the extreme cases where there is just one hypothesis and hence no ranking of hypotheses. This model allows the GAP hypothesis only as an alternative to other hypotheses about the internal structure of an NP and not as an alternative to other hypotheses at the phrasal level. Therefore, the model predicts that a GAP will be detectable in proportion to the expectation of a full noun phrase in that position. Fodor concludes that the lexical-expectation model accounts in impressive detail for the data she discusses.

3.4 Some Remarks about Perceived Processing Complexity. Before I discuss doubtful gaps in the context of deterministic parsing, I would like to say a few words about the concept of 'perceived processing complexity'. As we have just seen, Fodor relies on her intuitive judgements of processing complexity to choose among the various models of gap finding she discusses. There are two major problems with this approach, however.

It would seem that Fodor implicitly assumes that the human parser is of the backtracking variety. Although she is not explicit about this, it is implicit in her account of perceived processing complexity and in her terminology (e.g. the parser can be momentarily 'decoyed' by a doubtful gap). With this in mind consider such a well-known example of the garden path phenomenon as *The cotton clothing is made of grows in Mississippi.* Most people who come across this sentence (and ones similar to these) for the first time tend to experience a severe disruption of their parser as soon as they get to the word *grows*. The disruption is caused by one's initial failure to identify the fragment *clothing is made of* as a relative clause. What such examples hammer home rather forcefully is that a failure of the parsing process gives rise to a perception of extreme 'complexity' ('disorientation' might be more appropriate), not just to some slightly increased complexity.

Another problem with the assumption that the human sentence processor is a backtracking parser is that such parsers have a rather prominent characteristic: they backtrack all the time (as is well-known to anyone who has ever seen a trace of the parse of a Prolog program). It is hard to see how one could possibly discriminate between the added processing complexity of the sort caused by doubtful gaps, as in (13i), and the 'background' processing complexity of the parser's standard backtracking behaviour. One might object that a suitably restricted backtracking parser would only backtrack if it was really necessary. Although such an objection may seem justified, it raises two new problems: (i) it must be shown that the restrictions on the backtracking parser yield exactly the observed processing complexity (which seems far from trivial, to put it conservatively; anyone who has ever tried to tweak a backtracking parser into selective backtracking will acknowledge that such a task is extremely cumbersome); (ii) why is backtracking restricted in precisely these ways and why does the brain not exploit the advantages of full backtracking? In short, an account of perceived complexity judgements in terms of a backtracking parser would seem to require extensive justification.

4 A strictly deterministic solution to the doubtful gap problem: separating trace creation and coindexing

As I pointed out in the previous section, the parsing models Fodor discusses in her article are all of the backtracking type. And indeed, the existence of doubtful gaps does not seem to agree very well with strict determinism. A Marcus-type parser would appear to be a first-resort mechanism by definition. After all, when it is confronted with a doubtful gap, its strictly deterministic mode of operation forces it to decide what action to take before it can continue. Because of its very limited ability to look ahead and see what the right context looks like, rightward disambiguation is usually beyond its scope, which condemns it to a first-resort strategy and therefore to occasional errors.

There is a way, however, to avoid the conclusion that Marcus-type parsers are first-resort gap fillers by definition. So far we have (tacitly) assumed that creating a trace involves coindexing it with its antecedent. This is by no means a necessary assumption, however. Rizzi (1988) proposes to reinstate the notion of 'referential index' in its full glory and to 'restrict its use to cases in which a referential index is made legitimate by certain referential properties of the element bearing it' (Rizzi op. cit.: p.2). He goes on to suggest that the use of referential indices should be restricted to cases made legitimate by the following principle:

(16) A referential index must be licensed by a referential θ-role

This is very much the original *Aspects* approach to indices:

> 'Suppose that certain lexical items are designated as referential and that by a general convention each occurrence of a referential item is assigned a marker, say, an integer, as a feature [fn. omitted]. . . . The semantic component will then interpret two referential items as having the same reference just in case they are strictly identical - in particular, in case they have been assigned the same integer in the deep structure.'
> (Chomsky 1965:145-146)

I will not go into the details of Rizzi's paper, which is entirely devoted to an extensive recasting of the ECP, but will simply use the same basic idea about the status of referential indices for my own purposes.

In particular, let us assume that the parser cannot assign an index to a chain link unless it has *unambiguously* identified the θ-marked foot of the chain. Suppose furthermore that adjuncts and adjunct-operators carry an inherent θ-role[7]. Finally, assume that any element (including an empty operator) becomes invisible in the LF-component, unless it has been associated with a referential

[7]See Van de Koot (1990) for a detailed treatment of the analysis and interpretation of adjunct-operator chains in deterministic parsing.

index[8].

Chain construction could now proceeds as follows. Suppose the current node contains an overt argument-operator. Then the parser adjoins an *unindexed* trace to every following maximal projection until it detects a gap. We now have to consider two cases.

Case (i). Suppose the detected gap is a doubtless gap. Then the parser drops an *indexed* trace and aborts its chain algorithm. That is, it does not adjoin a trace to the next maximal projection. When the parser has come to the end of the input, it starts attaching completed constituents. Let us refer to this as the parser's 'backward mode'. If the parser is in backward mode, and it drops a completed constituent in the buffer that contains an indexed trace, then it coindexes that trace with the antecedent trace (or the overt operator, as the case may be) in the current node. This process continues until the parse tree is complete. The output will contain a chain in which all chain links are coindexed, as required.

Case (ii). The second case we have to consider arises when the detected gap is a doubtful gap. In this case the parser drops an *unindexed* trace, and continues to adjoin *unindexed* traces. We now have to consider three subcases: (a) no further gaps are detected, (b) a doubtless gap is detected, or (c) one or more additional doubtful gaps are detected. Of these only the first case is straightforward. Suppose no further gaps are detected. Then, when the parser switches into backward mode and revisits VP in which it found the first doubtful gap, the completed phrase in the first cell of the buffer will contain an unindexed trace. This provides unambiguous evidence that the doubtful gap is in fact the true gap and the parser now indexes the trace. The rest of the indexation process now proceeds as in case (i) above.

Subcases (b) and (c) form a dangerous combination. To see why, consider the following questions. First, suppose a sentence could really contain two or more doubtful gaps, then how would we represent the global chain ambiguity that would arise in such cases? Second, suppose the parser had indeed found one or more doubtful gaps. Suppose further that, when it switched to backward mode, it dropped an indexed trace for the most deeply embedded doubtful gap. Then an index would be passed up on every subsequent step. Then, by the time the parser revisited the VP which contained the one but most deeply embedded doubtful gap, it would have no way of determining whether the index that was passed up originated in a doubtful gap or a doubtless gap. The same ambiguity would arise if we took the option of not passing up an index. I.e. during the backward mode we would have no way of distinguishing subcase (b) from subcase (c).

Seeing that multiple doubtful gaps complicate the parsing problem still further, let us make the simplifying assumption that multiple doubtful gaps do not in fact exist. There are some apparent counterexamples to this claim, to be discussed in the next section, which all involve multiple gaps that are not clausemates. If we restrict attention to competing gaps that *are* clausemates, however, as in all of Fodor's examples (except (9), see below), then the

---

[8]Van de Koot (1990) assumes in addition that any element that delimits a range (i.e. any overt *wh*-element) is visible at LF. This additional assumption is crucial in accounting for the parser's robust behaviour under subjacency violations, but plays no role in the discussion that follows.

following observation seems to be correct: if a doubtful gap in the complement position of a verb is disconfirmed by the detection of a gap to its right, then that newly discovered gap will always be a doubtless gap, overtly marked by the presence of a preposition[9]. Cases like (9) above create special problems. The important thing to notice about sentences like these is that the gap in subject position is *not doubtful*. It *is* ambiguous (between a PRO or a trace)[10]. Sentences like (9) will yield to a strategy that is quite similar to that discussed under case (i) above. Upon encountering the ambiguous gap, the parser could postpone creating the CP and IP projections until it had parsed the rest of the sentence. During the rest of the parse it would continue to adjoin unindexed traces to every new maximal projection, so as to guarantee the presence of a local antecedent in case it detected a doubtless gap. By the time the parser returned to the ambiguous subject gap it would either have an unindexed or an indexed trace in the position adjoined to the complete VP in the first cell of the buffer, depending upon whether it had found a doubtless gap. This would give the parser the required information for disambiguation of the subject gap.

Suppose then that these problems can be dealt with satisfactorily. This will clear the way for the assumption that multiple doubtful gaps do not exist. Given this, we can ignore subcase (c), which involved two or more doubtful gaps. What about subcase (b)? Suppose that after having found a doubtful gap, the parser detects a doubtless gap. In that case, we will assume that it drops an indexed trace for the doubtless gap and aborts its chain algorithm. When it switches into 'backward mode', it takes care of the indexing of the chain links as before. When the the VP that contained the doubtful gap becomes current again, the parser does not index the trace for the doubtful gap, which will therefore be invisible at LF, as required[11].

---

[9]As is well-known, things become rather more complicated if the moved constituent is a PP, because that adds the PP-attachment ambiguity problem to our list of worries. I will abstract away from the PP-attachment problem, however, focussing attention instead on cases of *wh*-movement of NPs, such as the ones discussed by Fodor. Carlos Gussenhoven (personal communication) claims to have found substantial evidence that intonational information goes a long way towards disambiguating PP-attachment.

[10]There are in fact sentences quite similar to those in (9) which are globally ambiguous:

(i) who do you want to succeed
(ii) who do you want PRO to succeed
(iii) who do you want t to succeed

What we do not have in (i), however, is a globally ambiguous *chain*. Cases such as these are truly problematic for a deterministic parser, as indeed is any globally ambiguous sentence. How a deterministic parser settles upon one of the possible readings of an ambiguous sentence is a problem quite distinct from the one considered here and I will put it aside here.

[11]We must also assume that case is assigned to referential indices only, so as to avoid visibility of the trace at PF in violation of the Projection Principle. This seems straightforward.

5 Apparent cases of multiple doubtful gaps in complex sentences

Baart and Raaijmakers (1988), who seem to be unaware of Fodor's work on doubtful gaps, suggest that the Marcus parser could be supplemented with a last resort strategy for sentences like (17) below, which, they claim, have a doubtful gap in the matrix clause that can only be disambiguated by inspecting the parser's right context:

(17) (i) welke studenten vertelde hij (t) dat hij t uitgenodigd had
which students told he that he invited had
(ii) welke studenten vertelde hij (t) dat hij zich verslapen had
which students told he that he himself overslept had

We will shortly have reasons to reinterpret the data in (17), but for now let us accept them. Even though Baart & Raaijmakers present no detailed analysis of a chain algorithm, the idea they have in mind is clear. When the parser encounters the doubtful gap in the matrix clause, it does not immediately drop a trace, but instead builds the embedded clause first. Upon detection of a doubtless gap in the embedded clause, it builds a chain backward. If no such gap is detected, it drops a trace in the matrix clause when it has finished the embedded clause. Baart and Raaijmakers' proposal could perhaps be made to work technically and would not constitute an unnatural extension of the parser's abilities. But despite its initial attraction, backward chain formation suffers from problems that are the exact mirror image of the ones it was designed to solve. Consider the examples below:

(18) (i) to whom did Bill say that Frank believed Susan gave a book (t)
(ii) did Bill say that Frank believed Susan gave a book (t)

In (18i) chain formation will be in 'forward mode' all the way down to the variable. Therefore, when the parser encounters the doubtful gap, there is a local antecedent and no problem arises. But now consider (18ii). There is no antecedent, so forward chain formation does not take place. However, what happens when the parser detects the doubtful gap in the embedded clause? The problem is that at this point in the parse the Marcus parser has no way of knowing whether there is an antecedent for the gap it has detected. Therefore, it cannot resolve the ambiguity and halts, clearly a very undesirable result. In the Baart & Raaijmakers' examples no such ambiguity arises, because in all their examples the gap in the embedded clause happens to be a doubtless gap.

The dilemma for this sort chain algorithm may be summarized as follows. The deterministic approach to chain formation demands that ambiguities be resolved on the basis of strictly local information. However, the structure of the natural language problem is such that this demand cannot be met with a simple wait-and-see strategy (for it is the postponed creation of traces that gives rise to new problems). Note in particular that the ambiguity problem in (18ii) cannot be solved by increasing the accessibility of the parser's left context: a doubtful gap that lacks a local antecedent either has no antecedent at all, as in (18ii), or its antecedent can be arbitrarily far away:

(19) to whom did you promise (t) to tell Sharon that Carl believed that Sue said . . . that Bill would talk t

The chain algorithm developed in the previous section does not suffer from the same shortcoming as the Baart & Raaijmakers algorithm. The reason for this is that it does not do backward chain formation, but only backward chain indexing: trace *creation* is *always* antecedent-dependent. But the data in (17) and (18) did not yet provide the worst cases that a parser has to deal with. As I pointed out earlier, a case of multiple doubtful gaps would do our algorithm in as well. With cases like (19) under our noses, we do not have to look very far for the dreaded example:

(20) to whom did Bill promise (t) that he would give a book (t)

It would seem that with examples like this one we have a serious problem at our hands. Fodor (1978) gives a few examples that are similar to (20), but she does not seem to notice their very problematic nature:

(21) (i) who did Mary promise (t) that she would marry (t)
(ii) to whom did Father say (t) that he was planning to write (t)

Fodor writes:

> 'Many informants have agreed that as one reads through sentence [(21ii)], one's first intuition is that the *to whom* fits into the doubtful gap in the main clause, giving the analysis *Father said to whom that...* Then, as one continues through the sentence, its meaning changes with much the same subjective effect as a reversible visual figure such as the duck-rabbit, so that the *to whom* now fits into the doubtful gap in the second clause: *Father said that he was planning to write to whom.*'

If these intuitive judgements are correct, none of the models Fodor discusses accounts for them. An assumption that is implicit in Fodor's article is that the parser cannot postpone or avoid decisions about its input. It is this rigid behavior that underlies her account of the relative processing complexity of her examples. The parser can be momentarily 'decoyed', as she puts it, and then must undo some of its previous actions. On this view of things, the parser will also have to decide whether to assume a gap after *promise* and *say* or not. Assuming the best model Fodor presents, the lexical-expectation model, this means that, like all all other verbs with doubtful gaps, *promise* and *say* must be associated with a ranking of hypotheses. Hence, the parser either assumes a gap or it does not assume a gap. Whatever it decides, the non-preferred alternative reading will exhibit added processing complexity. However, this is not what we find with these sentences[12].

Going back to the data in (20) and (21), it seems reasonable to argue that the problem they pose is only apparent and due to a misanalysis of the

[12]The other option is to account for the judgements in terms of a parser that builds more than one analysis in parallel. This seems untenable, however, in the light of such phenomena as garden paths (see section 3.4 and Marcus op. cit. for details).

facts. Engdahl (1984) and Hudson (1984) analyze cases similar to (19)-(21) as involving parasitic gaps. Consider (22):

(22) which men$_i$ did the police warn t_i [$_{CP}O_i$ that they were about to arrest e_i]

We could analyze the data above analogously. Here are some examples:

(23) (i) to whom$_i$ did Bill promise t_i [O_i that he would give a book e_i]
(ii) who$_i$ did Mary promise t_i [O_i that she would marry e_i]
(iii) to whom$_i$ did Father say t_i [O_i that he was planning to write e_i]

On this view the gap in the matrix clause would in fact not be a doubtful gap. The judgements of these data vary from sentence to sentence and from individual to individual. The parasitic gap reading appears to be facilitated by (at least) two factors: the choice of matrix verb and absence of tense in the embedded clause, but it is the first of these which seems to have the greatest influence. My own judgements are that the Dutch facts completely parallel the English. Informants almost invariably return a judgement with either a gap in the matrix sentence and no gap in the embedded sentence or a parasitic gap reading.

What makes these data particularly striking, is that the embedded clause does not generally speaking constitute a barrier to movement. Therefore, the parasitic gap is not inside a syntactic island as is usually the case:

(24) (i) to whom did Bill promise his wife that he would give a book t
(ii) who did Mary promise her father that she would marry t
(iii) to whom did Father say to the linguist that he was planning to write t

Although the analysis of these sentences remains a matter of some concern, I see at present no reason to abandon the assumption that multiple doubtful gaps do not exist. In fact, the strong preference that many speakers seem to have for a doubtful gap reading of the sentences in (23) is suggestive. For a detailed treatment of parasitic gaps in the context of deterministic parsing I refer the reader to Van de Koot (to appear).

6 Conclusion

I have argued that the doubtful gap problem can be solved strictly deterministically, on condition that global chain ambiguities do not occur in natural languages. The proposed chain algorithm obeys Rizzi's (1988) principle governing the assignment of referential indices, in that trace indexing is dependent upon the unambiguous identification of the θ-marked foot of the chain of which the trace forms a link. The logical separation of trace creation and trace indexing, together with the assumption that only elements carrying a referential index are visible at LF, was shown to account for the data in a satisfactory way.

It can easily be shown that the chain algorithm has linear time complexity. The number of computational steps needed to build and coindex one trace is

some constant *k*. The amount of traces and operators the parser builds is linear in the amount of input words. Hence, the algorithm's complexity is *kn*. All in all, the conclusion is justified that the special structure of the chain formation problem does not warrant a computationally expensive method like backtracking.

References

Baart J. and S. Raaijmakers (1988). 'Dutch as a Deterministic Language'. In Coopmans, P. and A. Hulk eds., *Linguistics in the Netherlands 1988*. Dordrecht: Foris.

Berwick, R. and A. Weinberg (1984). *The Grammatical Basis of Linguistic Performance*. Cambridge, MA: MIT Press.

Chomsky, N. (1965). *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press.

Engdahl, E. (1984). 'Parasitic Gaps, Resumptive Pronouns, and Subject Extractions'. Ms., University of Wisconsin, Madison.

Fodor, J. (1978). 'Parsing Strategies and Constraints on Transformations'. *Linguistic Inquiry* 9, pp.427-473.

Hudson, R. (1984). 'Multiple (alias 'Parasitic') Gaps'. Ms., University College, London.

Koot, J. van de (1985). Review of *The Grammatical Basis of Linguistic Performance. Second Language Research* 1.1., pp.73-82.

Koot, J. van de (1987). 'On Explaining Subjacency'. In F. Beukema and P. Coopmans, eds., *Linguistics in the Netherlands 1987*, pp.121-129. Dordrecht: Foris.

Koot, J. van de (1990). *An Essay on Grammar-Parser Relations*. Doctoral dissertation. University of Utrecht.

Kuroda, S.-Y. (1986). 'Whether We Agree or Not: Rough Ideas about the Comparative Syntax of English and Japanese'. Ms. University of California at San Diego.

Marcus, M. (1980). *A Theory of Syntactic Recognition for Natural Language*. Cambridge, MA: MIT Press.

Rizzi, L. (1988). 'On the Status of Referential Indices'. Paper presented at the workshop 'The Chomskyan Turn', Tel-Aviv, Jerusalem.

Wanner, E., R. Kaplan, and S. Shiner (ms.). 'Garden Paths in Relative Clauses', unpublished paper, Harvard University, Cambridge, Massachusetts.

Wanner, E., and M. Maratsos (ms.). 'An Augmented Transition Network Model of Relative Clause Comprehension', unpublished paper, Harvard University, Cambridge, Massachusetts.

Wanner, E., and S. Shiner (ms.). 'Ambiguities in Relative Clauses', unpublished paper, Harvard University, Cambridge, Massachusetts.