Natural Language Engineering **5** (3): 271–300. Printed in the United Kingdom

© 1999 Cambridge University Press

Parsing with discontinuous phrases

ALLAN RAMSAY
Centre for Computational Linguistics, UMIST,
PO Box 88, Manchester M60 1QD, UK
e-mail: allan@ccl.umist.ac.uk

(*Received February 1999; revised November 1999*)

Abstract

Most parsing algorithms require phrases that are to be combined to be either contiguous or marked as being 'extraposed'. The assumption that phrases which are to be combined will be adjacent to one another supports rapid indexing mechanisms: the fact that in most languages items can turn up in unexpected locations cancels out much of the ensuing efficiency. The current paper shows how 'out of position' items can be incorporated directly. This leads to efficient parsing even when items turn up having been *right*-shifted, a state of affairs which makes Johnson and Kay's (1994) notion of 'sponsorship' of empty nodes inapplicable.

1 Chart parsing

Consider the following rather general statement of the Fundamental Rule of Chart Parsing (FRCP):

If X and Y are abutting structures and X is unsaturated and Y will help saturate it then merge them to form a new, possibly saturated, structure X' covering the combined spans of X and Y.

Of course, this has to be tightened up a little to ensure that X and Y are actually abutting in the required manner – that if X expects Y to be on its right then the end point of X and the start of Y must be identical, and *vice versa* if Y should be to the left of X; and it may have to be adapted to cope with grammars where items can be combined because one of them modifies the other as well as because one is an argument of the other. Nonetheless, this rather general statement characterises the standard operation of a chart-parser (Kay 1973; Kaplan 1973). The difference between using such a parser bottom-up or top-down lies in the source of incomplete edges – in a top-down parser, incomplete edges arise from considering rules which might be expanded to obtain desirable items, in a bottom-up one they arise from considering items that trigger rules by matching some element of the right-hand side.

In nearly all implementations of this notion, it is assumed that X and Y are contiguous. This seems like a reasonable assumption: by and large, phrases are made out of smaller contiguous parts. Consider (1):

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

1 *Jack and Jill went up the hill to fetch a pail of water.*

Almost any analysis of this would lead to a well-formed bracketing that looked something like

2 *((Jack and Jill) (went (up (the hill)) (to (fetch (a (pail (of water))))))).*

The labels associated with these phrases would vary from theory to theory, but the conclusion that (1) can be bracketed roughly as given is widely shared across grammatical frameworks.

When implementing the FRCP, you need somewhere to put the (saturated and unsaturated) items that you have constructed so far. The key notion behind chart parsing is that you can avoid repeating work that you have done already if you keep track of every phrase or part-phrase that you have ever considered. If you do that then judicious bookkeeping will ensure that you will notice any attempt to repeat earlier work before you actually start to do anything with it.

For this to work, you have to be able to search the database of previously constructed items to find (a) ones that are identical to the one you have just created, in order to avoid repeating earlier analyses, and (b) ones which can combine with the one you have just created, in order to make up new ones. For both these tasks, it makes sense to index phrases and part-phrases ('edges', in chart parsing terminology) on their start and end points.

To make this concrete, I shall consider an implementation for an HPSG-like treatment of English where an individual sign contains information about the structure and appearance of a given phrase in addition to the usual phonology and syn-sem (there are significant differences between this treatment and either Pollard and Sag (1988) or Pollard and Sag (1994): most of these differences, apart from the treatment of 'extraposition', are largely irrelevant here. What matters is that I am continuing to use the notion of a 'sign' as a repository of various kinds of information about a word or phrase.)

Within such a framework, the word *and* in (1) would give rise to a structure looking rather as follows[1]:

```
sign(structure(positions(start(1), end(2), text(and))),
     syn(nonfoot(head(cat(U), ...)),
         subcat(args([sign(structure(dir(+right, -left)),
                           syn(nonfoot(head(cat(U), ...),
                           meaning(...)),
                      sign(structure(dir(-right, +left)),
                           syn(nonfoot(head(cat(U), ...),
                           meaning(...)),
         foot(slash=defval(value([]), EM),
              focus=defval(value([]), EN),
```

[1] Upper case words in signs denote variables, with repetitions of such variables marking structure sharing/reentrancy, as in Prolog. So the fact that all three items in this sign have `U` as their `cat` indicates that they will all have the same basic category.

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

```
            wh=defval(value([]), EO))),
    meaning(...))
```

This describes an unsaturated phrase starting at position 1 (I count from 0) and ending at 2. The positional information is included inside the sign, rather than by embedding the sign within a process oriented structure such as an edge, because I am taking it that the structure is a description of *this* occurrence of the word. This is useful in situations where the actual position of a word within a sentence is significant. Suppose, for instance, that you believed that a WH-word could only introduce a question if it was the first word in the utterance. Under those circumstances, the grammar would have to have access to positional information of the kind encoded here within the sign. Similarly, the rule that says that certain classes of English modifier attach to the left of their targets if they are head final (*the quietly sleeping man*) and to the right if they are not (*the man sleeping quietly in the park*) (Williams 1981) depends on having positional information explicitly encoded within the sign. Bird and Klein's (1994) analysis of the phonological properties of a word in a given context encodes information about specific occurrences of a word within the corresponding sign in much the same way. Other examples will emerge in section 4.

This phrase is looking for two items – one, of unspecified major category `U` to its right (marked by the fact that this item's `direction` is `dir(+right, -left)`[2]); and another of the same unspecified type to its left. If it finds two such items it will produce something of the same type that incorporates them both.

I have removed a lot of detail from this description of the sign that arises from the presence of the word *and* in sentence (1). What matters here is that this sign contains positional information about itself (that it starts at 1 and ends at 2) and about its arguments (that the first should appear to the right and the second to the left: it is, in fact, an open question whether a conjunction should look for its rightward argument before it looks for the leftward one. In the example I am assuming that it is better to look right first, but it would be easy to accommodate the other order). This positional information is usually accommodated by refining the FRCP to

> If the end point of X is identical to the start point of Y, and X is looking to its right for an item of type Y', where Y' is a description which fits Y[3], then merge them to form a new, possibly saturated, structure X' which starts where X starts and ends where Y ends.
>
> If the start point of X is identical to the end point of Y, and X is looking to its left for an item of type Y', where Y' is a description which fits Y, then merge them to form a new, possibly saturated, structure X' which starts where Y starts and ends where X ends.

This version of the FRCP permits appropriate items to be combined if their start and end points are identical (with two versions being introduced to allow for

[2] The use of sets of polar-valued features as a mechanism for avoiding disjunction when manipulating feature sets is discussed in (Ramsay 1990).

[3] Exactly what is meant by saying that a description fits an item will vary from theory to theory. If we assume that the information we have about Y is complete, then Y' will fit Y if the information specified in Y' subsumes Y's description. If we allow the description of Y to be partial, then Y' will fit it if they unify.

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

bidirectionality (Steel and de Roeck 1987)). As such, it would work extremely well if phrases that needed to be combined always occurred in the place where they were expected. This, however, is not the case.

2 Extraposition and 'empty' items

Even in English, phrases can be wrenched from their canonical positions, to appear far away from where they were expected:

3 *I saw the man who your sister said she wanted me to introduce her to in the pub.*
4 *In the pub last night I saw your sister's boyfriend.*
5 *He said he believed with all his heart that she loved him.*
6 *She was, I believe, lying.*

Sentence (3) is a simple case of left-extraposition of a WH-marked pronoun to the boundary of an embedding clause in order to make a well-formed relative clause. In sentence (4) the VP-modifying phrases *In the pub* and *last night* have been left-shifted, either to make them thematic or simply in order to reduce the range of PP-attachment ambiguities. In sentence (5) the sentential complement *that she loved him* has been *right*-shifted from its canonical position, again in order to reduce ambiguity (the unshifted form, with *with all his heart* attached to the VP *believed that she loved him*, would have been *He said he believed that she loved him with all his heart*, which sounds extremely odd). Finally, sentence (6) seems to involve left *and* right extraposition of components of the clause *she was lying* from *I believe she was lying*.

The last of these is perhaps debatable, though I find it tempting. In any case, sentences (3)–(5) indicate that you cannot blithely rely on arguments and modifiers to turn up where you expect them. But if they are not going to turn up where they are supposed to, you have to abandon the assumption in the second version of the FRCP that you can use the start and end points of phrases to constrain what should be merged with what.

Most grammatical theories posit the existence of some sort of 'empty' items to cope with these cases. The assumption is that the second *to* in sentence (3) ought to have an NP to act as its argument. Since there isn't a real one we posit the existence of a 'trace' of one[4]. We then somehow carry the fact that we have introduced this item up through the structures that include the resulting PP until we find the word *who*, which can fill in the trace and which doesn't appear to be doing anything useful where it is. In unification grammars such as HPSG and GPSG (Gazdar, Klein, Pullum and Sag 1985) this information is carried on a feature traditionally called `slash`. In other frameworks it is carried in other ways, but the notion that we introduce a place-holder for a missing item at the point where it should have appeared, and then cancel it against something suitable which we encounter elsewhere, is widespread.

[4] The terminology for such items varies from theory to theory, but the basic idea is widespread.

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

The problem is that this notion undermines the simple nature of the FRCP. The usual solution involves adding two more cases, namely

If X is looking to its left or right for an item of type Y, then introduce a trace of one and merge them to form a new, possibly saturated, structure X' which starts where X starts and ends where X ends. The description of X' will include the fact it includes a trace of a Y (and so will any larger structure that contains X' up to the point where the trace is eliminated).

If X is a sentence containing anywhere within itself a trace, and Y is an adjacent item of the specified kind, then merge them to form a sentence which is *not* missing anything.

The first of these can lead to an explosion in the number of edges to be considered. As it stands there is nothing to stop you positing such phrases everywhere – every argument, every potential modifier. Johnson and Kay (1994) suggest that you should look for 'sponsors' for such items, i.e. for things which might play the role of Y in the second of these new cases. After all, if there isn't going to be anything to play this role then the trace will never be cancelled, which means that there will never be a well-formed sentence containing it.

This proposal, however, inevitably lets through cases which are destined to fail. The fact that you cannot predict whether the sponsor is actually going to be the item that is absorbed by the cancellation means that you will necessarily allow slashed items to be sponsored by phrases that eventually turn out to be irrelevant. You can lessen this risk by making the search for sponsors more constrained (e.g. by requiring there to be a tensed verb between a slashed item and its sponsor), but this makes the search for sponsors more expensive and is in any case still not foolproof. Worse than that, the potential for *right*-extraposition in cases like (5) makes it impossible to combine a general left→right strategy in the parser and a search for sponsors: the sponsor simply won't be there at the point when you need it.

3 Direct instantiation

The intuition underlying Johnson and Kay's suggestion prompts an alternative approach. Instead of using the existence of a sponsor to act as a control over whether it is worth introducing a trace for an extraposed item, why not just use it directly?

To see what this means, consider first a simple case of left-extraposition of a WH-pronoun:

7 *which he ate*

Suppose we are processing this with a head-corner parser (Bouma and van Noord 1993), which does as much as it can with each lexical item in turn before moving on. At the point when we consider '*ate*' we will have two NPs – one going from 0 to 1 for '*which*' and one from 1 to 2 for '*he*'. Consideration of *ate* will lead to the introduction of an edge which would make an S if two NPs, one accusative and one nominative, were found.

Using the extended version of the FRCP above, we would introduce a trace for the accusative NP, sponsored by *which*. We would then find the nominative one in

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

its correct place, i.e. immediately to the left of the verb, to produce a sentence with a slashed NP. Since this is adjacent to an appropriate NP we would cancel them, producing a well-formed sentence. Consideration of *ate* will lead to the introduction of an edge which would make an S if two NPs, one accusative and one nominative, were found.

Using the extended version of the FRCP above, we would introduce a trace for the accusative NP, sponsored by *which*. We would then find the nominative one in its correct place, i.e. immediately to the left of the verb, to produce a sentence with a slashed NP. Since this is adjacent to an appropriate NP we would cancel them, producing a well-formed sentence spanning all three words.

Alternatively, we could simply pick up *which* as the object of *ate* immediately. We would then have a VP containing the words *ate* and *which*. This VP is immediately preceded by a nominative NP: we can then therefore combine the two, producing a sentence which contains the three words *ate*, *which* and *he*.

The first approach uses positional information as a constraint on which phrases can combine with one another: the fact that phrases can appear 'out of position' means that we have to introduce empty items and hope for something which will fill them in to turn up as we proceed. Under the second approach we are guaranteed to have a filler, since we will have incorporated it immediately. We seem, however, to have lost the ability to index items effectively.

This problem can be fixed by considering the items that a phrase contains rather than concentrating on its start and end points. Consider the following version of the FRCP:

If X and Y are non-overlapping structures and X is unsaturated and Y will help saturate it then merge them to form a new, possibly saturated, structure X' covering the combined spans of X and Y.

We need to put some flesh on the notion of non-overlapping here. I will represent the span of an edge by a bit-string, where there is a 1 in position N of this string if the edge includes word N and a 0 otherwise. In the case of (7), the edges corresponding to the single words *which*, *he* and *ate* would have spans of 001, 010 and 100 respectively, with the leftmost word having a 1 in the rightmost position in the corresponding bit-string, and subsequent words having bits set in increasingly left-shifted positions[5]. Then we can test whether two items are disjoint simply by taking the bitwise-& of their spans: if this is 0 then they have no words in common. Likewise the combined span of two items is just the bitwise-OR of their individual spans.

So in the example, we can combine *ate* and *which* because `001 /\ 100` is 0 (I use `/\` to denote bitwise-&, and `\/` for bitwise-OR). The span of the combination is `101`. This phrase can in turn be combined with *he*, since `101 /\ 010` is also 0, leaving us with a saturated sentence whose span of `111` indicates that it covers the entire text.

[5] This may seem rather backwards, but it is easier to manipulate spans if they are reversed in this way.

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

4 Well-formedness

The simple notion outlined above will clearly work for simple cases such as (7). What will happen when we try non-trivial cases like (3)?

There are two related questions here:

- How will we stop it over-generating?
- How will we stop it taking forever?

If we can answer the first then we will have a solution to the second. If we can ensure that only well-formed analyses emerge from this approach to extraposition then we will have put constraints on the possibilities that need to be considered, namely the well-formed ones. It will turn out that although the constraints are most clearly stated with respect to completed analyses, they have consequences which can be detected at intermediate stages so that flawed partial analyses can be ruled out early. As a consequence, the number of edges that get added to the chart is about as small as could be.

The constraints come in two forms: one deals with linear precedence (LP) relations, the other with constraints on where extraposed items may go to. The first kind are easy to describe.

4.1 Linear precedence constraints

As we build a structure, we keep track of the items that it is made of – its 'daughters'. We keep track of their general syntactic properties, and we keep track of the position where they were found. We can then easily check whether local order constraints are violated. For instance, in English no local sub-tree which is WH-marked can follow one which is not. This is why *which he ate* is acceptable whilst *he ate which* is not[6].

Local constraints of this kind, which strongly resemble the LP-rules of GPSG and constraints such as the 'case filter' of GB (Chomsky 1981), can be given declarative descriptions. The rule described above, for instance, can be described as follows (the notation `F@A` here means the value of the feature `F` in the object `A`):

```
{A, B}: +wh@A & -wh@B => start@A < start@B
```

where the prefix {`A, B`} says that this rule applies to any pair of items `A` and `B` which are local subtrees of some structure.

It is easy to check such rules at the point in the parsing process where new items are constructed. In particular, you can assume that either `A` or `B` is the local subtree which is being attached at this stage, which avoids checking that existing subtrees still obey the rules.

As it happens, HPSG deals with much of the work performed by such constraints

[6] And why *I want to know who ate what* is also acceptable. The point here is that the occurrence of *what* in the embedded clause *who ate what* does not need to be extraposed, since the other, leftward, daughter of *ate* in this phrase is also WH-marked. Note that the headless relative *who ate what* is not itself a WH-marked item, and hence does not violate this rule within the larger sentence (and the pronouns *who* and *what* are not local subtrees of the main sentence, and hence also escape).

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

by lexical rules. These rules, which fill in the details of subcategorisation frames, are generally applied *before* parsing starts, so that there is no need to check them at this point. Nonetheless, in frameworks where such local order constraints are required they can easily be checked as soon as arguments are assimilated, as outlined above.

4.2 Constraints on extraposition

The constraints we need to impose on which items may be extraposed, and where they may be extraposed to, are more interesting. There are, in general, two reasons for shifting an item from its canonical position:

1. For discourse reasons: to make it more prominent, to put it into a 'thematic' position. This is clear in cases like

   8 *I enjoyed the soup, but the main course I thought was awful.*

   If I am right that sentence (6) involves extraposition then this also seems to be a case where the goal is to mark some parts of the utterance as more significant than others (in this case, the goal is to downplay the significance of *I believe* rather than to direct attention to *she was lying*, but the effect is the same).
2. For processing reasons:
   (a) You may want to shift some item which carries a marker denoting a global property of a phrase to the boundary of that phrase (usually the left boundary). The obvious example of this is the extraction of WH-marked phrases, in order to give an immediate indication that the embedding clause is itself WH-marked.
   (b) You can often reduce ambiguity by shifting items around. Left extraposition of *in the pub* and *last night* in sentence (4), and right extraposition of *that she loved him* in sentence (5), both serve this purpose. You can also use right extraposition to reduce processing load by postponing analysis of complex structures. This can be seen in 'heavy NP shifts' of the kind that occur with particle-verbs, e.g. *He gave up the job he had worked so hard to obtain* vs. *Why did he give it up?*.

In either case, the extraposed item nearly always ends up at a sentence boundary (the positioning of clitics in French may be an exception to this).

The easiest way to state the constraints on extraposition involves defining the notion of a 'compact' phrase.

A phrase is **compact** if it contains all the words that appear between its extreme left and right ends.

This notion is crucial: fortunately it is very easy to check whether a phrase is compact or not. If `I` and `J` are the extreme, not natural, start and end points of a

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

phrase and SPAN is the bit string encoding its span, then the phrase is compact if and only if SPAN equals ((1 << (J-I))-1) << I[7].

To make use of this check, we need to extend the positional information that we include in an edge. We need the extreme left and right end points; we need the span; and we also need the 'natural' left and right end points – the boundaries of the phrase if we *exclude* the extraposed items. Consider, for instance, the relative clause

9 *who he gave it to*

Processing of sentence (9) will proceed as follows.

1. The verb *gave* will take *it* as its direct object, to form a +compact phrase *gave it*, whose natural and extreme starts are both 2, and whose natural and extreme ends are both 4. This phrase would have a sign that looked like

```
sign(structure(positions(start(2),
                         end(4),
                         span(01100),
                         +compact,
                         xstart(2),
                         xend(4))),
     syn(nonfoot(head(cat([xbar(+v, -n)]), ...), ...),
         subcat(args(["PP", "NP"]))),
     meaning(...))
```

   The argument list contains a PP and an NP: the PP, which is the next item to be found, is the indirect object[8], and the NP is the subject. The fact that the indirect object should appear to the right and the subject to the left is encoded within their descriptions. The span encodes the fact that this phrase includes the words starting at positions 2 and 3.
2. The preposition *to* will combine with *who* to form a -compact phrase whose extreme start is 0 – the start of *to* – and whose natural start is 4 – the start of the head. The extreme and natural ends are both 5. The sign looks like

```
sign(structure(positions(start(4),
                         end(5),
                         span(10001),
                         -compact,
                         xstart(0),
                         xend(5))),
     syn(nonfoot(head(cat(pp), ...), ...),
```

[7] 1 << (J-I) left shifts 1 by J-I. Suppose we were considering a phrase which started at 2 and ended at 4 (i.e. it covers two words, one going from 2 to 3 and the other from 3 to 4). Then 1 << (J-I) would be 1 << 2, i.e. the bit string 100. If we subtract 1 from this, we get the string 11, and if we then left shift this by 2, we get 1100, in other words a string with 1's in positions 2 and 3, corresponding to the start positions of the words in the current phrase.

[8] There are good reasons for treating this item as a case-marked NP rather than as a PP. I have left it as a PP here just to keep the presentation clear.

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

```
        subcat(args([]))),
    meaning(...))
```

The span shows that it includes the words starting at positions 0 and 5. The fact that it contains gaps is the reason why this phrase is -compact.

3. This will be absorbed by the unsaturated phrase *gave it* to produce a -compact phrase whose extreme start is 0 and whose natural start is 2. The extreme and natural ends are again 5. The resulting sign is

```
sign(structure(positions(start(2),
                         end(5),
                         span(11101),
                         -compact,
                         xstart(0),
                         xend(5))),
    syn(nonfoot(head(cat([xbar(+v, -n)]), ...), ...),
        subcat(args(["NP"]))),
    meaning(...))
```

Again, the natural start and end are the boundaries of the compact group of words containing the head. The span is the bitwise-OR of the spans of the contributing phrases, since it includes all the words that they do.

4. Finally, this will incorporate *he*, to form a saturated compact phrase (since it contains every word from 0 to 5) whose extreme and natural starts are both 0, whose extreme and natural ends are both 5, and whose span is 11111.

4.2.1 English

The general rule for extraposition in English is then

If X is a compact clausal phrase then any extraposed items must be at its extreme left or right boundaries.

This covers most cases, though we clearly need some constraints on which items can be extraposed at all.

Right extraposition

It is particularly difficult to control right extraposition. The fact that right extraposition is generally there to make processing easier means that it is in fact very difficult to put static constraints on it. How could you tell whether something was going to lead to an attachment ambiguity until you looked at it? How could you tell whether an NP was heavy enough to be worth shifting until you had compared it with the items it was going to be shifted past? Consider, for instance, sentence (10):

10 *He built on that site one of the largest and most unattractive houses of the entire Renaissance period.*

What we have here is right extraposition of the object of a perfectly ordinary

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

transitive verb. If even innocent items like this can be right extraposed, there seems to be little we can do in advance to constrain right extraposition. The best we can do, apparently, is to look at cases where something does appear to have been moved to the right and see whether we are happy about it.

The situation is not, in fact, quite as bad as that. We do know that some arguments will never be right extraposed. The subjects of most verbs won't be, for instance (though the obvious analysis of things like *At the front of the bus sat an old man* does seem to involve right extraposition of the subject). We also know that some items will never turn up out of position – that pronouns, for instance, never get right extraposed.

11 *He gave it up.*

12 * *He gave up it.*

It looks as though the canonical order here puts the object before the particle, with the object being allowed to undergo right extraposition, as in:

13 *He gave up the job which he had worked so hard for.*

The problem with sentence (12) is that the pronoun *it* is not allowed to fill a right extraposed position.

To state these constraints we need somewhere to store information about whether some item can move/has moved, and if so whether the movement is to the right or the left. We therefore extend the positional information encoded in the description of a sign to carry this information. The following extract from the description of the pronoun *it* shows that this item is banned from being used as an argument unless it appears in the canonical position for that argument:

```
sign(structure(extraposition(dir(-moved, -right, -left))),
     syn(nonfoot(head(cat([xbar(-v, +n), noun, +pronoun]),
                      agree(W, X, third(+sing, -plural)))),
     meaning(...))
```

The fact that this is marked as `-moved` says that any context in which *it* appears must be one in which it has not been moved (and just to make sure the marking `-right, -left` also says it must not have been moved to the right or to the left).

We can use the same set of features to indicate whether the context allows some item to be left or right shifted. The following extract from the lexical entry for *builds* says that this word requires two arguments, one playing the role of `object` and the other the role of `agent`. The object is the first thing to be searched for. It ought to appear to the right of the verb (encoded in the fact that it has to match `dir(+right, -left)`, but as far as *builds* is concerned there is no reason to believe that it won't be shifted either to the right (as in sentence (10)) or to the left (in something like *the house which he built*). This is encoded by the fact that the cluster of features under `extraposition` is completely unspecified. The agent, on the other hand, is supposed to appear to the left of the verb (`dir(-right, +left)`): it is allowed to shift, but not to the right (`extraposition(dir(DW, -right, DY))`).

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

```
sign(structure(...),
     syn(...,
         subcat(args([sign(structure(dir(+right, -left),
                                     extraposition(dir(CC, CD, CE))),
                           syn(...),
                           meaning(uses((role(object)), ...))),
                      sign(structure(dir(-right, +left),
                                     extraposition(dir(DW, -right, DY)),
                           syn(...),
                           meaning(uses(role(agent)), ...))])),
     meaning(...)
```

We use the `extraposition` features to specify whether a given item is allowed out of position and whether a given argument or modifier is allowed to appear out of position; and, after an argument or modifier has been absorbed, to record whether in fact it *did* appear in a non-canonical location, and if so whether the shift was to the left or the right.

Left extraposition

What about left extraposed items? The key question here concerns the permissible surface orders in sentences with multiple left extrapositions. Why, for instance, is sentence (14) better than sentence (15)?

14 *the man who in January I received a letter from*

15 * *the man in January who I received a letter from*

In both cases *who* and *in January* have both been left shifted. Sentence (14), while clumsy, is perfectly acceptable: sentence (15) is not (at least not with the intended meaning: it does have an analysis which would have been more apparent if the example had been *the man in Moscow who I received a letter from*). The problem with sentence (15) does not, however, arise from the order in which the extracted items are found. It is, rather, a consequence of the rule from section 4.1 which says that WH-marked items must precede non-WH marked ones in surface order. To see this, consider

16 *Do you remember the time when John you hated but Mary seemed to be OK?*

which is also acceptable, albeit extremely contrived, whereas

17 *Do you remember the time John when you hated but Mary seemed to be OK?*

is just impossible. In each of sentences (14) and (16), a modifier and an argument have both been left shifted. In sentence (14) the argument has been left shifted further than the modifier, whereas in sentence (16) it is the other way round. In both cases, however, the WH-marked item is the one that has gone furthest, in accordance with the rule from section 4.1.

We could simply use the rule as given to weed out unacceptable cases. This would,

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

however, lead to the construction of large numbers of partial analyses which were doomed to failure but which would clutter up the database (as we will see later, complexity of the parsing algorithm has a term which depends upon the square of the number of edges, so avoiding unnecessary edges is well worth doing whenever possible).

Fortunately, the rule has a number of corollaries which can also be expressed as LP rules, and which can be used to rule out potential partial analyses well before we get to compact clauses (note that in sentence (3) we don't get a compact clause until we have dealt with the whole of *who your sister said she wanted me to introduce her to*). Two obvious cases are:

- No compact non-clausal item may contain left extraposed constituents.
- If the subject of a clause is left shifted then it must be WH-marked.

The first of these simply reflects the fact that, in English, extraposed items have to be located at clause boundaries. If we have a compact non-clause containing a left shifted item then that item cannot be at a clause boundary, which is clearly not allowed. The second is a consequence of the fact that, again in English, the subject is normally the leftmost constituent of the sentence. Why might you want to shift something which is supposed to be at the left hand edge even further to the left? Either because it is WH-marked, in a case like *Who did you say wanted to do it?*; or to get it into a thematic position. But the only reason why the subject would *not* be in such a position would be because something else had already been shifted there, in which case it would make no sense to try to shift the subject back over it.

The first of these rules is a direct consequence of the general principle. The second arises from interactions between this principle and other rules of the grammar. Given any particular grammar, there will be a range of local constraints arising from such interactions. These derived local constraints are in a sense theorems of the grammar, rather than axioms. They are, nonetheless, extremely useful when we are trying to use the grammar to obtain structural analyses.

4.2.2 German: main and subordinate clauses

It has been widely argued that the best way to account for the existence of 'verb-second' constructions in Germanic languages is to posit an underlying verb initial structure, and to require declarative main clauses to have exactly one constituent topicalised (Uszkoreit 1987; Diderichsen 1966). The argument can be extended to subordinate clauses, where *every* constituent has to be topicalised (Ramsay and Schäler 1997).

Consider, for instance, the ditransitive verb *gab*(*gave*). This typically requires three arguments, roughly playing the role of agent, object and recipient. The basic facts relating to the order of these arguments are as follows:

- The agent will be allocated the surface role of subject, and will be in nominative case. The object will be allocated the surface role of direct object, and will be accusative marked. The recipient will be allocated the surface role of

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

second object, and will be either morphologically marked as dative, or will be case-marked by an appropriate preposition.

- The relative order of these arguments is unconstrained. Specific choices will have discourse consequences, e.g. if the subject is pronominalised then it will almost certainly be the leftmost of the three, and there are also wider constraints, for instance that in relative clauses the relative pronoun *must* be the leftmost. Nonetheless, all six orders (*agent-object-recipient, agent-recipient-object, object-agent-recipient, object-recipient-agent, recipient-object-agent* and *recipient-agent-object*) are possible in certain circumstances.
- The verb may appear as the second item in a sentence (in simple main clause declarative sentences), at the beginning (in polar questions) or at the end (in subordinate clauses). When the verb appears at the second position, the item, the item to its left may be either one of the arguments or a modifier.

The approach to displaced items presented here is particularly appropriate for this. If you take any largely bottom-up strategy (such as head-corner parsing, which is in many ways the optimal strategy) you will find yourself having to introduce traces for all possible constituents (modifiers *and* arguments). You have to be prepared for any argument or none to be left shifted to obtain the verb-second forms (you have to be prepared for none of the arguments to be shifted, since it could be a modifier that takes the crucial position); and for subordinate clauses you have to be prepared either for every constituent to be left shifted, or (equivalently) for there to be an alternative subcategorisation frame in which all the arguments appear to the left of the verb and all modifiers are left shifted.

Under the proposed regime you simply look for arguments and modifiers wherever you can find them. If they all turn up to the right of the verb in their canonical positions, you must be looking at a polar query. If exactly one turns up to the left and none are WH-marked, you're looking at a declarative main clause. If they all turn up on the left you're looking at a subordinate clause. As with English, if any are WH-marked then they must be at the extreme left. The same general rule as for English will apply, namely that extraposed items must appear at the extreme left or right end of the sentence. The details of why items get extraposed in German are slightly different, and hence the LP-rules which block ill-formed cases are slightly different, but the general mechanism goes through unchanged.

4.2.3 French: clitic placement

It is also plausible that the positioning of clitic pronouns in French can best be accounted for in terms of left extraposition. The crucial data is given below:

18 *J'ai donné mon vélo à Bill.* ('I gave my bike to Bill' – direct and indirect objects are full NPs, with the indirect object case-marked by the preposition *à*)

19 *Je l'ai donné à Bill.* ('I gave it to Bill' – direct object pronominalised and left-shifted)

20 *Je lui ai donné mon vélo.* ('I gave him my bike' – indirect object pronominalised to dative marked form and left shifted)

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

21 *Vous me l'avez donné.* ('You gave me it' – both objects pronominalised and left-shifted. Note that the indirect object *precedes* the direct object)

22 * *Vous moi l'avez donné.* (Unacceptable because the wrong form of the first person pronoun has been used. Is the problem that the *moi* has the wrong case marking?)

23 *Vous le lui avez donné.* ('You gave him it' – both objects pronominalised and left-shifted. This time the indirect object *follows* the direct object)

24 * *Vous le m'avez donné.* (Variant of sentence (21) with the same order as sentence (23), showing that the decision as to whether the indirect object precedes or follows the direct object is not a free choice.)

25 * *Vous lui l'avez donné.* (Similarly using the (21) word order for (23))

26 *Vous ne me l'avez pas donné.* ('You didn't give me it' – just to show the relative positions of the negation marker and the clitic pronouns)

27 *Donnez-le-moi.* ('Give me it'. The pronouns are not left-shifted in the imperative, and this time *moi* does appear to be the indirect object, which suggests that the problem with sentence (22) was not the case marking on *moi*)

28 * *Donnez-le-me.* (Using the form of the pronoun that appeared in (21) is unacceptable when the pronoun is not left-shifted)

29 *Donnez-le-lui.* ('Give him it'. The relative positions of the two pronouns are the same here as in sentence (23))

30 *Donnez-moi-le.* Sentences ((30) and (31) again show that the relative order of the pronouns is not unconstrained [9])

31 * *Donnez-lui-le.*

32 *Ne lui le donnez pas.* ('Don't give him it' – when the imperative is negated then the pronouns do get left-shifted)

Sentence (18) lays out the canonical word order for a simple ditransitive verb, with the direct object and indirect object following the verb and the subject preceding it. Sentences (19) and (20) show what happens when one of the objects is pronominalised, namely that the pronoun takes up a position *between* the subject and the head verb (where I assume that the dominant auxiliary is the head of the clause: note that this means that *peux* in *Je peux vous en donner* must be a substantive verb with an infinitive sentential complement with a raised subject. This is a perfectly reasonable thing for it to be). Sentences (21)–(32) show that when *both* objects are pronominalised the order in which they appear is tightly constrained but is not easily related to the roles they are playing: in sentence (21), for instance, the indirect object precedes the direct one, whereas in sentence (23) it is the direct one that comes first.

[9] Though sentences (30) and (31) are, apparently, permissible, in some dialects.

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

The first move is to specify the circumstances under which a clitic pronoun must be left shifted. That can be done fairly easily:

If a non-subject argument is pronominalised in any non-imperative, or a negated imperative, sentence then it must be left extraposed.

If a non-subject argument is pronominalised in an unnegated imperative sentence then it must not be left extraposed.

As it stands this explains some of the facts in sentences (18)–(32). There are two further steps to be taken. Firstly, we have to provide descriptions of individual pronouns which encode the facts about whether they can or must appear in left- or right-shifted positions. Different pronouns display different behaviours: it therefore seems sensible to include the information about how a pronoun will behave in its lexical description. *me*, for instance, can only be used in left-shifted positions, and *moi* cannot be so used, whereas *le*, when viewed as a pronoun, can be used either way. So the lexical entries for *me*, *moi* and *le* are as follow:

```
sign(structure(extraposition(dir(+moved, -right, +left)),
               text("me")),
     syn(nonfoot(head(cat([xbar(-v, +n), noun, +pronoun]),
                      agree(first(+sing, -plural), W, X))),
     meaning(...))

sign(structure(extraposition(dir(-moved, -right, -left)),
               text("moi")),
     syn(nonfoot(head(cat([xbar(-v, +n), noun, +pronoun]),
                      agree(first(+sing, -plural), W, X)))),
     meaning(...))

sign(structure(extraposition(dir(_A, -right, +left)),
               text("le")),
     syn(nonfoot(head(cat([xbar(-v, +n), noun, +pronoun]),
                      agree(W, X, third(+sing, -plural)))),
     meaning(...))
```

The entry for *me* says that it must be shifted (`+shifted`), and that the shift must be to the left (`-right, +left`). *moi* must not be shifted (`-shifted`). The value for `shifted` for *le* is left unspecified, so that it will match either `+shifted` or `-shifted`, indicating that it can appear either shifted or not; but if it is shifted, then it must be *left*-shifted (`-right, +left`). This explains why sentences (22) and (28) are unacceptable whereas we can use *le* in both sentences (23) and (31).

Finally we have to explain why sentences (24), (25), (30) and (31) are unacceptable. As noted above, the problem cannot arise from the assignment of thematic roles to the various pronominalised arguments, since the indirect object precedes the direct one in sentence (21) and follows it in sentence (23). Consequently, it is also impossible to explain it in terms of their relative positions in the canonical order, since these are determined by their thematic roles. It seems that we simply need

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

a partial ordering of clitic pronouns which is determined by some characteristic property of the individual pronouns (possibly some sort of phonological constraint) rather than a universal rule related to thematic roles or case marking. This partial order can clearly be stated as a set of LP-rules, and hence can be checked whenever a new item is constructed:

```
{A, B}: text@A <-> le & text@B <-> lui => start@A < start@B
...
```

In all these cases the subject must also be left extraposed. All we need to do, therefore, to accommodate the order of the pronouns in sentences (18)–(31) is to require that the subject is the leftmost item. This can easily be expressed by an LP-rule saying that any nominative marked local subtree must precede any other.

```
{A, B}: +nom@A & -nom@B => start@A < start@B
```

The only way that this can happen in the cases where there is a clitic pronoun is if the subject is left shifted, but we have allowed for that and hence there is no problem.

There is a great deal more to be said about French clitics, of course (Kayne 1975), but I am not the person to say it and this is not the place. The discussion above does, however, show that we can at least begin the discussion within the current framework, and that a treatment in terms of direct incorporation of displaced constituents is possible.

5 Internal and external syntax

The approach outlined above will work fine for cases where the 'out of position' item is concretely realised somewhere in the sentence. This is not, unfortunately, always the case. Consider, for instance, sentence (33):

33 *I saw the man you were talking about in the pub last night.*

This includes a reduced relative clause. You can cope with this in frameworks where you introduce a trace for an NP in the position where it *ought* to be and then pick it up later. The trick here is to recognise that a sentence with an uncancelled slashed item can function as a relative clause even if you never find the WH-pronoun. Furthermore, you *will* find a sponsor for this item, namely the NP that immediately precedes it. As it happens, this sponsor will not later be used to cancel the gap, but that doesn't matter. All you need is the presence of some sponsor, and a recognition that a sentence with an uncancelled slash can be used as a relative clause.

Under the approach proposed here this will not work. We have to find all the constituents before we can claim that we have a sentence. In general we do not worry about the fact that some of these constituents may turn up in unexpected places, but we do have to find them all. What will happen when we try to process sentence (33)?

The phrase *the man you were talking about* will be analysed as a topicalised sentence. Perfectly straightforward: *about* will pick up *the man* as its complement,

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

to make a `-compact` PP. This will then function as a modifier on *talking*, which will be combined with *you* and *were* to make a `+compact` clause containing a left shifted item. The problem is that we have to realise that this can be seen as an NP containing a relative clause.

The key here is the phrases 'can function as' and 'can be seen as' in the preceding discussion. It often happens that items which *look* like Xs can be *used* wherever Ys are required. Obvious cases in English include the use of appropriately marked WH-clauses as NPs (in cases like '*I don't know much about art but I know what I like*', where *'what I like*' is acting as an NP complement to '*know*'); the use of present participle VPs as NPs, as in '*he concluded the banquet by eating the owl*', where '*eating the owl*' is an NP complement of '*by*'; and present participle verbs acting as nouns, as in '*The killing of Sister George*', where '*killing*' must be a noun, since otherwise neither '*the*' nor '*of Sister George*' could combine with it.

This notion of the difference between what something looks like and what it can be used for has been explored at some length in tagmemics (Pike 1982), but it has been largely ignored within computational linguistics (though recent developments in 'construction grammar' (Kay 1997) are aimed at least partially at these issues). Consideration of the examples above indicates that we do still need it. The traditional approach involves introducing a distinction between the internal and external syntax of a phrase, something like:

The **internal syntax** of a phrase is the description that is obtained by looking at the items that make it up, notably its head. The head of *eating the owl*, for instance, is *eating*. The phrase is thus, internally, a VP – something whose head is a verb and which has had all its arguments apart from its subject supplied.

The **external syntax** of a phrase corresponds to its distributional pattern. Externally *eating the owl* can appear in (almost) every location where an NP can appear. It can occur as the complement of certain prepositions, e.g. *He had a glass of cider after eating the owl, I am terrified of eating the owl, …*, and it can appear in subject or object position for many verbs which would normally require NPs in these positions, for instance *Eating the owl made him very ill, he greatly regretted eating the owl.*

Malouf (1996) suggests that these cases can be dealt with by using lexical rules. This seems entirely reasonable for nominal gerunds (cases where a verb, usually a present participle, takes on the role of a simple noun). Thus, in *the killing of Sister George*, the word *killing* looks like a present participle verb, but has a number of properties that are more characteristic of nouns. These properties are both internal and external:

Internal: the nominal form inherits at least some of the subcat list, namely that the verb *kill* requires a direct object. However, for the nominal form, this object has to be case-marked by the preposition *of*. This is characteristic of transitive nouns, not of verbs.

External: the phrase *killing of Sister George* displays all the external characteristics of an $\overline{\mathrm{N}}$ – it requires a specifier (of exactly the same kind as is required by any other $\overline{\mathrm{N}}$), it can be modified by an adjective (*the unexpected killing of Sister George*), and any phrase that it heads will go anywhere an NP is required.

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

It seems entirely appropriate to use a lexical rule for this, e.g.

```
[verb, prespart, args=ARGS] ==> [ngerund, thirdsing, args=ARGS]
ngerund < noun
```

The first part says that when you see a present participle verb with `ARGS` as its subcat list you can introduce a nominal gerund with the same subcat list. The second part says that `ngerund` is a sub-type of `noun` (and hence is `-specified`), and hence that you can use the lexical rules that describe the case-marking of noun arguments to note that the direct object has to be marked by *of*.

Malouf extends this to deal with verbal gerunds (*Killing Sister George was a bad mistake*), with a rule like

```
[verb, prespart, args=ARGS] ==> [vgerund, thirdsing, args=ARGS]
vgerund < noun, vgerund < verbal
```

where `vgerund` is a subtype of both `verbal` and `noun`. This has a number of consequences:

- It becomes very hard to explain where it inherits the information about the case-marking of its arguments from. If it's `verbal`, you would expect the direct object to be accusative (as it is), but if it's nounlike then you would expect it to be marked by *of*.
- It appears that the lexical rule introduces a separate lexical item, rather than refining the description of the original. If that is the case, then in an example like *My writing this paper about parsing strategies was aided by the referees' comments* we will find two lexical entries arising from *writing*. Each of these will require the direct object to be found. The way this will be found is identical in each case: but the work will be repeated. Since the phrase *this paper about parsing strategies* contains four NPs (*this*, *this paper*, *this paper about parsing* and *this paper about parsing strategies*) then each of the entries for *writing* will produce phrases covering *writing this*, *writing this paper*, *writing this paper about parsing* and *writing this paper about parsing strategies*.

We can reinvent the more traditional approach by allowing for 'post-lexical rules'. A post-lexical rule introduces an alternative view of a phrasal item, in the same way that a lexical rule introduces an alternative view of a lexical item. So we would have a rule of the form

```
[vp, prespart] ==> [nominal, thirdsing]
```

where the type `nominal` has no value given for `specified`, and hence can appear in contexts where an NP is required (*Painting his daughter gave Brown great pleasure*) and in ones where an $\overline{\text{N}}$ is required (*Brown/Brown's painting his daughter really irritated the class.*).

The use of post-lexical rules avoids the repeated work mentioned above, since it is not until we have found the VP *writing this paper about parsing strategies* that we even contemplate treating it as some kind of nominal[10], and it also explains why the

[10] We will, of course, still get all four, and rightly so: but we will avoid the repeated derivations on the way.

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

case-marking follows the pattern for verbs: because we are actually only thinking about a verb.

It also allows us to cover the other cases where something which doesn't look like an NP can be used in contexts where one is required. It is, for instance, most implausible that you could deal with the fact that headless relatives are distributionally a subclass of NPs by introducing a lexical rule that said that a verb could be seen as an unsaturated NP if one or more of its arguments were WH-marked. The relevant post-lexical rule, however, is simplicity itself:

```
[s, +tensed, +wh] ==> [headlessrel]
headlessrel < NP
```

Similar arguments apply to reduced relative clauses. You can't have a lexical rule which explains why a clause with an NP slashed out of it can be used as a modifier for $\overline{\text{N}}$s. There is really no alternative to introducing something like

```
[s, +tensed, -wh, slash={NP}] ==> [reducedrel]
reducedrel < nn_modifier
```

In the present context, this rule is unavailable, since we do not have an explicit list of slashed items waiting to be cancelled. We can, however, introduce a rule which says that a sentence with a topicalised NP can, itself, be seen as an NP containing a reduced relative clause.

```
[s, +tensed, -wh, topicalised={NP}] ==> [np]
```

Deriving the semantics of the resulting NP requires a certain amount of deconstruction of the given sentence, but the syntactic analysis goes through just fine – we will analyse the first seven words of *the man I wanted you to meet has just left* as a topicalised S, and then apply the post-lexical rule to re-view them as an NP, and hence get an appropriate analysis of the whole sentence.

6 Complexity and performance

The proof of the pudding is in the eating. Clearly, we *can* deal with extraposition by looking for the displaced items directly, rather than positing empty items and later cancelling them against appropriate items which we find at clause boundaries. Is it a good idea?

There are two ways of trying to answer this. We can try to give an analysis of the relevant complexity of the two approaches; and we can simply try them out and see which is quicker.

6.1 Complexity

The standard complexity results for chart parsing apply to situations where you have a large grammar, fairly simple lexical entries and an assumption that phrases will be compact. This is clearly inappropriate here. In order to get to an analysis that will help us evaluate the performance of the algorithm described in section 3 I will first

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

recast these results in terms of a head-corner chart parser for a simple categorial grammar. I will then show that the move to a more HPSG-like grammar has very little effect. Finally, I will consider what happens when we consider a grammar that allows for extraposition: the consequences of this final move are rather dramatic, at least in principle, and I will comment on the fact that in practice the situation does not seem as bad as these results suggest.

6.1.1 Complexity: monadic categorial grammar

Consider a simple categorial grammar with monadic basic categories, say `S`, `NP` and N, and the standard rules of combination (see Wood (1993) for a discussion of such grammars):

```
X ==> X/Y, Y
X ==> Y, X\Y
```

Note that I am not allowing type-raising, which is the usual mechanism within categorial grammar for dealing with extraposition. I want to consider the complexity of this approach to parsing *without* extraposition to start with.

Suppose we use this grammar with a simple chart parser that operates as follows:

1. Initialise the chart with edges for each lexical entry (including multiple entries for ambiguous words). To make this concrete, the edge that we would obtain if the third word in the sentence under consideration was *gives*, whose type is `((S\NP)/NP)/NP`, would look like

   ```
   edge(2, 3, ((S\NP)/NP)/NP)
   ```

2. If you find a pair of edges of the form

   ```
   edge(I, J, X/Y)
   ```

   and

   ```
   edge(J, K, Y)
   ```

   then create a new one of the form

   ```
   edge(I, K, X)
   ```

   (and similarly for left-seeking edges)
3. If the new edge is identical to one you've already got then throw it away. Otherwise store it in the chart and go to 2.

Fig. 1. Head-corner chart parser for basic categorial grammar.

It is well known that a simple categorial grammar of the kind described is equivalent to a context-free grammar. The contribution that the number of rules in a context-free grammar makes to the complexity of the parsing process manifests itself here in terms of the number of unsaturated edges after step 1 (this parameter is often ignored in complexity analyses of parsing algorithms, but it is a major determinant in the time taken for parsing).

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

Performing step 1 itself may, of course, be a fairly complex process (e.g. if obtaining the description of an item involves extensive use of lexical rules), but it will be linear in the length of the input sentence. Processing a single word requires you to (i) match the surface form against a stem and a set of affixes, possibly utilising a set of morphotactic rules to handle changes in surface form that result from choosing a particular affix, and (ii) apply any relevant lexical rules (which may lead to the generation of multiple forms). Neither of these steps depends on the length of the sentence, so the worst case is that initialising the chart for an N word sentence will take $N \times L \times K$, where L is the maximum time for matching the surface form for any form of any word in the dictionary and K is the maximum time taken to apply the lexical rules for any form of any word in the dictionary. In any case, as the concrete performance figures in section 6.2 indicate, it is not a major factor in the implementation described here.

What happens as we cycle through steps 2 and 3? Consider a typical edge, say `edge(I, J, X/Y)`. The only things that can result from using this are edges of the form `edge(I, K, X)`, where `K` is greater than `J`. There can only be `N-J` distinct edges of this form, where `N` is the length of the sentence.

Furthermore, there is a limit to how many times we can extend a partial edge. We can define the *rank* of a lexical item as the number of items that it requires in order to become saturated. Consider, for instance a preposition which is going to be used as a VP modifier. The type of such an item is usually given as `((S\NP)\(S\NP))/NP` – something which will combine with an `NP` to its right to produce something which will combine with a VP (i.e. an `S` missing an `NP`) to its left to produce a new VP. This will combine with three items – an `NP` to its right, followed by an `(S\NP)` to its left, followed by an `NP` to its left, to produce an `S`, and hence has a rank of 3. In any categorial grammar of the kind described, there is some lexical item whose rank is maximal. I will take the maximal rank of any item in the lexicon to be the rank of the grammar, ρ [11].

The existence of items which subcategorise for items which are themselves unsaturated, and which inherit all or part of the subcat list(s) of their arguments, complicates matters somewhat. The word *and* in *John smokes before and after lunch*, for instance, will combine with *before* and *after* to produce something which is itself of rank 3, so that we have to give *and* a rank of 2 + the maximal fixed rank of any items in the lexicon. Nonetheless, the notion of the rank of a grammar is well-defined and provides a useful tool for analysing the complexity of such grammars.

Suppose, then, that we have a sentence of length N for which step 1 has given rise to E initial edges, of which P are unsaturated. Then the maximum number of edges that could possibly be added to the chart is $\Sigma_{i=1}^{i=\rho} P \times (N - \rho)$, which is less than $\rho \times N \times P$. This would arise if and only if every initial unsaturated edge required

[11] Since categorial descriptions of this kind encode much the same information as is provided in the rules of a context-free grammar, you might expect a similar notion to be useful in considering parsing algorithms for CFGs. It turns out that because any CFG is weakly equivalent to a grammar with no more than two elements on the right-hand side of any rule, it is harder to use this notion for analysing the complexity of parsing with a CFG.

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

ρ items to be supplied to saturate it, and items of exactly the right kind spanning every possible subsequence of the sentence were present.

6.1.2 HPSG

HPSG differs from the skeletal categorial grammar of section 6.1.1 in many ways. The crucial points for the present discussion are as follows:

- Categorial grammar only allows for one way for items to combine, namely where one is an argument of the other. HPSG extends this in a number of ways: firstly, there are different ways for arguments to be assimilated (described by Schemas I–IV of Pollard and Sag (1994)). More importantly, a distinction is drawn between head-argument pairs and head-adjunct pairs.
- Subcat lists and the constraints on what an adjunct can combine with are now specified in terms of types, where a type is described by a (possibly disjunctive) set of constraints on feature:value pairs (Kasper and Rounds 1986; Kasper 1987; Gazdar, Pullum, Carpenter, Klein, Hukari and Levine 1988).

Pollard (1988, p 398) notes that '... the SUBCAT feature is close to being a notational variant of the slash notation in Steedman's system'. The introduction of head-adjunct combinations forces us to change the set of categorial principles slightly. The existing ones become

```
X'(args=T') ==> X(args=[right(Y) | T]), Y'
X'(args=T') ==> Y', X(args=[left(Y) | T])
```

and new ones are introduced to cope with head-adjunct combination:

```
X'' ==> X, Y(args=[], mod=left(X'))
X'' ==> Y(args=[], mod=right(X')), X
```

In the first rule, the unsaturated item `X` requires something of type `Y` to its right. The item `Y'` that appears in that position may belong to a subtype of `Y`, in which case the process of assimilating it may refine what is known about `X` and about the items on its subcat list, so that the resulting item `X'(args=T')` may well belong to a subtype of `X(args=T)`. Similar remarks apply to the other rules: note that in the second pair, the result `X''` of the combination will belong to a subtype of both `X`, the modified item, and `X'`, the type of item that `Y` can modify. In all four cases, however, the arguments above about the *number* of items required by a given adjunct or unsaturated item go through unchanged.

6.1.3 Free word order

Much the same analysis will go through if we allow for completely free word and phrase order. To simplify the presentation I will use an unordered categorial grammar where `X-Y` denotes an `X` that is missing a `Y` *anywhere* in the sentence. The parsing algorithm is given in figure 2.

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

1. Initialise the chart with edges for each lexical entry (including multiple entries for ambiguous words, and entries resulting from the application of lexical rules). To make this concrete, the edge that we would obtain if the third word in the sentence under consideration was '*gives*', whose real type is `(SP\NP)/NP)/NP`, would look like

   ```
   edge(100, ((S-NP)-NP)-NP)
   ```

   This says that we have an item which will produce an `S` if three `NP`s are found *anywhere* in the sentence. This is obviously not English – Finnish, perhaps. Note that where a standard chart parser stores information about an edge starts and finishes, this one just stores the edge's *span*, as discussed in section 4.2
2. If you find a pair of edges of the form

   ```
   edge(SPAN0, X-Y)
   ```

   and

   ```
   edge(SPAN1, Y)
   ```

   where `SPAN0 /\ SPAN1` is 0 then create a new edge

   ```
   edge(SPAN0 \/ SPAN1, X)
   ```

3. Check that the new item satisfies the LP-rules: these are stated as sets of rules of the form

   ```
   {A, B}: some condition on A and B => start@A < start@B
   ```

   They are checked by letting one of `A` and `B` be `Y`, and letting the other range over daughters of `X`.
4. If the new edge is identical to one you've already got then throw it away. Otherwise go to 2.
5. If the new edge triggers any post-lexical rules, add the required new edges.

Fig. 2. Chart parser for grammar with extraposition, LP-rules and post-lexical rules.

The major difference between the complexity of this algorithm and the previous one is that a partial edge that spans J items can now give rise to 2^{N-J} new edges. In the earlier case we knew where the edges that a given partial edge was to combine with would start (or end, for left-seeking edges), and we knew that they had to be compact. So for a right-seeking edge ending at word J we only had to consider edges that went from J to $J+1$, or from J to $J+2$, or ...or J to N. So there were only $N-J$ possibilities. This time we don't know where the target edge starts, and we don't know that it is compact. So now we have to consider all subsets of $\{1,\ldots,N\} \setminus \{\text{words already covered by the unsaturated edge}\}$.

Fortunately, we do still know that there is a bound on the number of items that an edge can combine with before becoming saturated, so we do still know that the maximum number of edges there can conceivably be in the chart is $\Sigma_{i=1}^{i=\rho}(P \times 2^{N-i})$. Thus the maximum number of edges in the chart is less than $\rho \times P \times 2^{N-1}$. So: parsing with completely free word order is a great deal more complex than parsing with fixed word order – hardly a surprising result.

The situation is, however, even worse than it seems. When we create a new edge, we have to see what existing edges it might combine with. For the simple case,

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

without extraposition, each edge had to be compared with all the edges already at the relevant point (all the edges starting where this one ends for right-seeking edges, and all those ending where this one starts for left-seeking ones). If we store edges in starts-at and ends-at tables, we only need compare a new edge with the entries in the relevant row of the appropriate table: this is certainly less than the entire collection of edges. If, as suggested above, we follow a left→right breadth-first strategy, then this row will actually be empty for any right-seeking edge!

Once we allow extraposition, there is clearly no indication of where the item we are looking for will be found. There is therefore no point in indexing edges according to their start and end points. Every new edge will have to be compared with every existing one. We are therefore potentially forced to perform $\Sigma_{i=1}^{i=F}(i-1)$ comparisons, where F is the number of edges at the end of the parsing process: this is $F \times (F-1)$, from which the dominant term is F^2. In the worst case F is $o(P \times 2^{N-1})$, so that the number of comparisons is $o((P \times 2^{N-1})^2)$, i.e. $o(P^2 \times 2^{2(N-1)})$.

This is potentially dreadful: quadratic in the number of unsaturated edges produced directly from lexical items, exponential in *twice* the length of the sentence. The results of section 6.2 indicate that the parser actually performs quite well. How can this be?

The analysis of the complexity of parsing with free word order assumes that *any* word or phrase can turn up in *any* position. As noted in section 4, this is not in general true. Only certain items can be extraposed (e.g. only compact arguments and VP-modifiers); they can only be extraposed under certain circumstances (WH-marked arguments can never be right shifted, non-clitic French pronouns can never be left shifted, etc.); and, most importantly, they can only be extraposed to certain locations (usually sentence boundaries).

This means that although the basic algorithm allows for very large numbers of edges with discontinuous constituents, not all that many actually turn up. In the analysis of

34 *Jack and Jill went up the hill to fetch the pail of water which they had left there.*

only six of the 129 edges created actually contain displaced items. So although the complexity of parsing with *unrestricted* extraposition looks as though it could be appalling, in practice very few extrapositions get hypothesised, so that the number of edges that are actually constructed is often closer to the $o(P \times N)$ of section 6.1.1 than the $o(P \times 2^{N-1})$ of section 6.1.3. As shown in section 6.1.1, there can be at most $\rho \times N \times P$ compact edges: if most edges are indeed compact, then the extra complexity that arises from introducing some non-compact ones does not become overwhelming. The constraints which rule out most of the spurious analyses are embodied in the LP-rules which are checked at step 4 of the algorithm in figure 2. The time taken to check these for a given edge depends only on the edge itself, so they do not affect the complexity of the whole process.

We still have to compare nearly every edge that is constructed with all its predecessors, since most of the constraints apply *after* we've found a potential match. This means that we cannot take much advantage of starts-at and ends-at tables, with the result that the number of comparisons to be made is still $o(F^2)$. But

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

at least F itself is generally better behaved than the worst case scenario of section 6.1.3 suggests.

6.2 Performance

In the end, however, what really matters is how long it takes. It is difficult to get a clear idea of how effective some component of a parser is by quoting parsing times, since it is very hard to isolate the effects of the scope of the grammar [12] and of the efficiency of other parts of the parser. Still, if I'm going to argue for a specific approach to parsing I should at least quote some figures. After all, if it took three days to parse *the cat sat on the mat* then it would not be interesting, no matter how persuasive the rest of the argument was. So here are a few examples:

3 *I saw the man who your sister said she wanted me to introduce her to in the pub.*

	T0 (secs)	P	Total (secs)	Edges	Analyses	Packed edges
Packed	0.21	15	1.24	114	8	5
Not packed	0.20	16	2.15	201	8	–

T0 here is the time taken to initialise the chart. P is the number of partial edges in the chart after initialisation (see section 6.1).

35 *Jack and Jill went up the hill to fetch the pail of water which they had stolen from the old man who lives in the park.*

	T0 (secs)	P	Total (secs)	Edges	Analyses	Packed edges
Packed	0.27	17	6.1	295	14	24
Not packed	0.25	17	12.5	484	14	–

10 *He built on that site one of the largest and most unattractive houses of the entire Renaissance period.*

	T0 (secs)	P	Total (secs)	Edges	Analyses	Packed edges
Packed	0.33	10	1.61	148	8	4
Not packed	0.31	10	4.8	329	8	–

The times here were obtained using compiled Sicstus Prolog on a Pentium 266 processor, and are elapsed times rather than CPU times. The 'Not Packed' rows

[12] If the grammar is large then parsing is likely to take longer, since there will be more locally feasible possibilities to investigate, even if none come to fruition: in particular, the admission of sentences with *right* extraposition will have fairly drastic effects on performance. The grammar I am using here produces appropriate analyses for all the English examples in the paper apart from sentence (6).

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

arose from allowing analyses that were internally different but externally identical to proceed as separate analyses (i.e. from failing to take advantage of the fact that the underlying parser uses a chart). The extra analyses in sentence (3) all result from attachment ambiguities. In sentence (35) the presence of the conjunction *and* inevitably leads to the creation of alternatives where either *Jack and Jill* or just *Jill* is the subject of some sentence. This compounds the attachment ambiguities associated with the various modifiers.

There are a number of points worth noting about these tables:

- The initialisation time covers lexical lookup, the expansion of lexical rules, and insertion of the initial lexical edges into the chart. The most important point about it is that it forms a very small proportion of the overall time for the parse – it hardly matters what the theoretical complexity is if the elapsed time is as small as this. I don't know why the initialisation time for sentence (10) is greater than for (35), which is longer and also involves more lexical ambiguity. The initialisation time on the 'Not Packed' rows is generally slightly smaller than for the corresponding 'Packed' ones: this is because checking that you haven't already got something that matches the edge you've just constructed takes time itself.
- As expected, speed does not depend linearly upon the number of edges. In the second case, for instance, the number of edges has gone up by a factor of 1.6 whereas the time has increased by 2.0. This results from a combination of the extra burden for the Prolog system in maintaining such a large database (each sign is an extremely bulky object) and the fact that we are performing roughly $o(E^2)$ comparisons between edges, where E is the number of edges. But at least it doesn't seem to increase at anything like the rate suggested by the complexity analysis of section 6.1.3.
- The number of edges saved by packing greatly exceeds the number of edges that we have to pack. In the first case we save 87 edges by packing 5, in the second we save 189 by packing 24, and in the third we save 181 by packing 4! This is as it should be, but it is good to know that it's true.
- Given that every phrase results from combining two smaller ones, the absolute minimum number of edges involved in obtaining an analysis of an N word sentence is $(2 \times N) - 1$. If there is a single lexical item which gives rise to M readings then this can go up to $M \times ((2 \times N) - 1)$. In view of the number of local ambiguities in sentences (3) and (35), only some of which manifest themselves as packed edges (even fewer lead to global alternatives), a total of 114 edges for a 19 word sentence and 295 for a 27 word one suggests that we are not creating all that many avoidable edges.
- Compilation[13] makes very little difference to the overall performance. It seems

[13] Lexical descriptions are preprocessed so that conjunctive type specifications are converted directly into Prolog terms and hence can be dealt with by unification, much as in ProFIT (Erbach 1995), so that any improvements that can be obtained by compilation should show up in the initial processing of lexical rules, and in steps 3 and 4 of the figure 2 algorithm which make use of LP-rules and post-lexical rules.

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

likely that this is because a very large part of the work performed involves asserting and retrieving database entries, where compilation has little impact.

7 Conclusions

I have outlined an approach to parsing in situations where items can appear 'out of position' which involves retrieving those items directly, rather than positing empty items and using feature percolation to propagate the fact that something is missing up to the point where it is found. This has both theoretical and practical advantages. As Sag (1997) argues, the presence of invisible linguistic items which have no surface manifestations is theoretically unattractive: if we can't see them, how do we know they exist? Sag proposes instead to deal with this phenomenon by introducing lexical rules which extend the sets of argument lists associated with individual items. Roughly speaking he proposes that a simple transitive verb like *builds* should lead to the production of two signs: one with the usual list of two NPs, and one with a list of one NP – the subject – and a marker saying that the other argument *has already been found*. In other words, rather than positing a single lexical sign, requiring two NPs to saturate it, and then immediately cancelling one of them against an invisible empty item, he posits two lexical signs. This may be marginally more appealing in that he no longer needs to consider the existence of empty items in his ontology. It hardly seems likely that it will make much difference to the complexity of the parsing problem. Indeed, since this happens during the processing of lexical rules it is very hard to see how you could take advantage of Johnson and Kay's (1994) notion of sponsorship, unless the processing of lexical rules and the general parsing process are very carefully interwoven. In contrast the approach outlined above does away with the need for empty categories without introducing a multiplicity of initial edges. The advantages to be gained by looking for sponsors are obtained immediately by using what would have been sponsors directly. This works even for right extraposition, which is not amenable to treatments involving sponsorship.

Bunt (1991) proposes an approach to a related set of problems involving insertion of unexpected material, roughly as in sentence (2) or in

36 *This is a better movie than I expected.*

where the modifier *better than I expected* seems to have become intertwined with its target *movie*. His approach involves explicitly noting points where inserted material may appear, using an extended version of GPSG's ID/LP format (Gazdar *et al.* 1985), and then using an extended set of metarules for picking up things that have been embedded in this way. It seems likely that this approach would require a very substantial increase in the size of the grammar, given the number of places where unexpected material may be found. In any case, Bunt notes that '... the verification of the DPSG adjacency-sequence relation requires a more elaborate bookkeeping' (more elaborate, that is, than the usual indexing mechanisms employed in chart parsers). The approach outlined in the current paper is more general than Bunt's proposal, and the bookkeeping is as close as possible to that of a normal chart parser.

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

The complexity analysis of section 6.1 is not encouraging. The performance figures of section 6.2, however, indicate that this approach *is* promising. It should be noted that LP-rules described in section 4 are essential: if you do not do effective filtering then the performance is roughly what you would expect from the complexity analysis.

References

Bird, S. and Klein, E. (1994) Phonological analysis in typed feature systems. *Computational Linguistics*, **20**(3): 455–491.

Bouma, G. and van Noord, G. (1993) Head-driven parsing for lexicalist grammars: Experimental results. In: S Krauwer, M Moortgat and L des Tombes, editors, *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics*, Utrecht. Association for Computational Linguistics.

Bunt, H. (1991) Parsing with discontinuous phrase structure grammar. In: M Tomita, editor, *Current Issues in Parsing Technology*, pp. 49–63. Kluwer Academic.

Chomsky, N. (1981) *Lectures on Government and Binding*. Foris Publications, Dordrecht.

Diderichsen, P. (1966) *Sætningsleddene og deres stilling – tredive år efter*. Helhed og Struktur – udvalgte sprogvidenskabelige afhandlinger, København.

Erbach, G. (1995) Profit: Prolog with features, inheritance and templates. *7th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 180–187. Dublin, Ireland.

Gazdar, G., Klein, E., Pullum, G. K. and Sag, I. (1985) *Generalised Phrase Structure Grammar*. Basil Blackwell, Oxford.

Gazdar, G., Pullum, G. K., Carpenter, R., Klein, E., Hukari, T. E. and Levine, R. D (1988) Category structures. *Computational Linguistics*, **14**(1): 1–19.

Johnson, M. and Kay, M. (1994) Parsing and empty nodes. *Computational Linguistics*, **20**(2): 289–300.

Kaplan, R. M. (1973) A general syntactic processor. In: R. Rustin, editor, *Natural Language Processing*, pp. 93–241. Algorithmics Press.

Kasper, R. (1987) A unification method for disjunctive feature descriptions. *ACL Proceedings, 25th Annual Meeting*, pp. 235–242. Association for Computational Linguistics.

Kasper, R. and Rounds, W. (1986) A logical semantics for feature structures. *ACL Proceedings, 24th Annual Meeting*, pp. 257–266. Association for Computational Linguistics.

Kay, M. (1973) The MIND system. In: R. Rustin, editor, *Natural Language Processing*, pp. 155–188. Algorithmics Press.

Kay, P. (1997) *Words and the Grammar of Content*. CSLI, Stanford.

Kayne, R. (1975) *French Syntax*. MIT Press.

Malouf, R. (1996) A constructional approach to english verbal gerunds. In: *Proceedings of the 22nd Annual Meeting of the Berkeley Linguistics Society*, Marseille, France. URL `http://hpsg.stanford.edu/rob/papers/`.

Pike, K. L. (1982) *Linguistics Concepts: an Introduction to Tagmemics*. University of Nebraska Press.

Pollard, C. J. (1988) Categorial grammar and phrase strucure grammar: An excursion on the syntax-semantics frontier. In: R. T. Oehrle, E. Bach and D. Wheeler, editors, *Categorial Grammars and Natural Language Structures*, pp. 391–416. Reidel.

Pollard, C. J. and Sag, I. A. (1988) *An Information Based Approach to Syntax and Semantics: Vol 1 Fundamentals*. Chicago University Press.

Pollard, C. J. and Sag, I. A. (1994) *Head-driven Phrase Structure Grammar*. Chicago University Press.

Ramsay, A. M. (1990) Disjunction without tears. *Computational Linguistics*, **16**(3): 171–174.

Ramsay, A. M. and Schäler, R. (1997) Case and word order in English and German. In: R. Mitkov and N. Nicolov, editors, *Recent Advances in Natural Language Processing*. Benjamin.

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press

Sag, I. A. (1997) English relative clause constructions. *J. Linguistics*, **33**(2): 431–484.
Steel, S. and de Roeck, A. (1987) Bidirectional chart parsing. *Advances in Artificial Intelligence (Proceedings of AISB-87)*, pp. 223–235. Wiley.
Uszkoreit, H. (1987) *Word Order and Constituent Structure in German.* CSLI, Stanford.
Williams, E. (1981) On the notions 'lexically related' and 'head of a word'. *Linguistic Inquiry*, **12**: 254–274.
Wood, M. M. (1993) *Categorial Grammars.* Routledge.

https://doi.org/10.1017/S1351324900002242 Published online by Cambridge University Press