Adjuncts and the Processing of Lexical Rules

Gertjan van Noord and Gosse Bouma

BCN RUG Groningen
{vannoord,gosse}@let.rug.nl

Abstract

The standard HPSG analysis of Germanic verb clusters can not explain the observed narrow-scope readings of adjuncts in such verb clusters.

We present an extension of the HPSG analysis that accounts for the systematic ambiguity of the scope of adjuncts in verb cluster constructions, by treating adjuncts as members of the subcat list. The extension uses powerful recursive lexical rules, implemented as complex constraints. We show how 'delayed evaluation' techniques from constraint-logic programming can be used to process such lexical rules.

1 Problem Description

1.1 Dutch Verb Clusters

Consider the following Dutch subordinate sentences.

(1) dat Arie wil slapen
that Arie wants to-sleep

(2) dat Arie Bob wil slaan
that Arie Bob wants to-hit
that Arie wants to hit Bob

(3) * dat Arie Bob wil slapen
that Arie Bob wants to-sleep
that Arie wants to sleep Bob

(4) * dat Arie wil Bob slaan

(5) dat Arie Bob cadeautjes wil geven
that Arie Bob presents want to-give
that Arie wants to give presents to Bob

(6) * dat Arie Bob wil cadeautjes geven
dat Arie wil Bob cadeautjes geven

(7) dat Arie Bob zou moeten kunnen willen kussen
that Arie Bob should must can want to-kiss
that Arie should be able to want to kiss Bob

The examples 1-3 indicate that in Dutch the arguments of a main verb can be realized to the left of an intervening auxiliary verb, such as a modal verb. Furthermore the sentences in 4-6 indicate that in such constructions the arguments must be realized to the left of the auxiliary verbs. In 7 it is illustrated that there can be any number of auxiliaries.

$$stem(\begin{bmatrix} \textsc{verbal} \\ sem|nuc|qfsoa : \begin{bmatrix} \textsc{want-soa} \\ arg1 : \text{Subj} \\ soa\text{-}arg : \text{Obj}\end{bmatrix} \\ subj|sem : \begin{bmatrix} index : \text{SjIx}\end{bmatrix}_{\text{Subj}} \\ sc : \langle \begin{bmatrix}\textsc{verbal}\\ sem : \text{Obj} \\ subj|sem|index : \text{SjIx} \\ sc : \text{A}\end{bmatrix} | \text{A}\rangle \end{bmatrix}).$$

Figure 1: The modal auxiliary 'wil'.

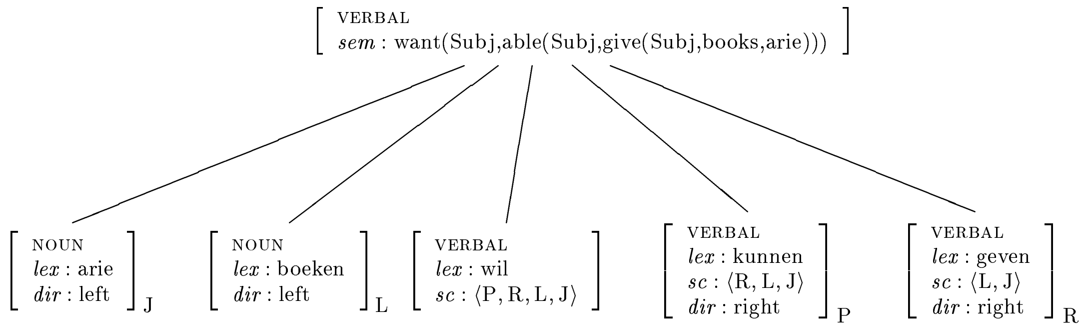

Figure 2: The parse tree for the verb-phrase 'arie boeken wil kunnen geven'.

1.2 The HPSG analysis of verb-clusters

The now standard analysis within HPSG of such verb-clusters is based on ideas from Categorial Grammar (cf. for example (Moortgat, 1988)) and defined within the HPSG framework by (Hinrichs and Nakazawa, 1989). In this analysis auxiliary verbs subcategorize for an unsaturated verb-phrase and for the complements that are not yet realized by this verb-phrase. In other words, the arguments of the embedded verb-phrase are inherited by the auxiliary.

For example, the auxiliary 'wil' might be defined as in figure 1. If we assume an application rule that produces flat vp-structures, then we obtain the derivation in figure 2 for the infinite verb-phrase

(8) ... Arie boeken wil kunnen geven

1.3 Problems with the scope of adjuncts

A major problem that this analysis faces is the possibillity of narrow-scope readings in the case of adjuncts. For example, the following Dutch subordinate sentences are all systematically ambiguous between a wide-scope reading (adjunct modifies the event introduced by the auxiliary) or a narrow-scope reading (adjunct modifes the event introduced by the main verb).

(9) dat Arie vandaag Bob wil slaan
that Arie today Bob want to-hit
that Arie wants to hit Bob today

(10) dat Arie het artikel op tijd probeerde op te sturen
that Arie the article on time tried to send
that Arie tried to send the article in time

$$\begin{bmatrix}\text{VERBAL}\\ sc:\mathrm{P}\cdot\mathrm{S}\\ sem:\mathrm{Sem}_0\end{bmatrix} \Longrightarrow \begin{bmatrix}\text{VERBAL}\\ sc:\mathrm{P}\cdot\langle \begin{bmatrix}\text{ADVERBIAL}\\ mod: \begin{bmatrix}\text{MOD}\\ arg:\mathrm{Sem}_0\\ val:\mathrm{Sem}\end{bmatrix}\end{bmatrix}\rangle\cdot\mathrm{S}\\ sem:\mathrm{Sem}\end{bmatrix}$$

Figure 3: A lexical rule that adds a single adjunct to the subcat list of a verb. In the case of n adjuncts the rule applies n times.

(11) dat Arie Bob de vrouwen met een verrekijker zag bekijken
that Arie Bob the women with the telescope saw look-at
that Arie saw Bob looking at the women with the telescope

Firstly note that the treatment of adjuncts as presented in (Pollard and Sag, in press), cannot be maintained as it simply fails to derive any of these sentences because the introduction of adjuncts is only possible as sisters of saturated elements. The fact that arguments and adjuncts can come interspersed (at least in languages such as Dutch and German) is not accounted for.

A straightforward solution to this problem is presented in (Kasper, in preparation). Here adjuncts and arguments are all sisters to a head. The arguments should satisfy the subcat requirements of this head – the adjuncts modify the semantics of the head (via a recursively defined adjuncts principle).

The main problem for this treatment of adjuncts is that it cannot explain the narrow-scope readings observed above. If adjuncts modify the head of the phrase they are part of then we will only obtain the wide-scope readings.

If we assume, on the other hand, that adjuncts *are* on the subcat list, then we will obtain both readings straightforwardly. In the narrow-scope case the adjunct is on the subcat list of the embedded verb, and then inherited by the matrix verb. In the wide-scope case the adjunct simply is on the subcat list of the matrix verb. In the next section we present a treatment of adjuncts in which each adjunct is subcategorized for. By means of lexical rules we are able to obtain the effect that there can be any number of adjuncts. We also sketch how the semantics of modification might be defined.

2 Adjuncts as Arguments

2.1 Adding adjuncts

The previous section presented an argument that VP modifiers are selected for by the verb. Note that this is in line with earlier analyses of adjuncts in HPSG (Pollard and Sag, 1987) which where abandoned as it was unclear how the semantic contribution of adjuncts could be defined.

Here we propose a solution in which adjuncts are members of the subcat list, just like ordinary arguments. The difference between arguments and adjuncts is that adjuncts are 'added' to a subcat list by a lexical rule that operates recursively.[1] Such a lexical rule might for example be stated as in figure 3.

Note that in this rule the construction of the semantics of a modified verb-phrase is still taken care of by a *mod* feature on the adjunct, containing a *val* and *arg* attribute. The *arg* attribute is unified with the 'incoming' semantics of the verb-phrase without the adjunct. The *val* attribute is the resulting semantics of the verb-phrase including the adjunct. This allows the following treatment of the semantics of modification [2], cf. figure 4.

[1] cf. (Miller, 1992) for a similar suggestions concerning French.
[2] inspired by (Kasper, in preparation)

$$\begin{bmatrix} \textsc{restr\_adverbial} & \\ mod : & \begin{bmatrix} arg|nuc : & \begin{bmatrix} qfsoa : \mathrm{Q} \\ restr : \mathrm{R} \end{bmatrix} \\ val|nuc : & \begin{bmatrix} qfsoa : \mathrm{Q} \\ restr : \langle \mathrm{R}_0|\mathrm{R} \rangle \end{bmatrix} \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} \textsc{op\_adverbial} & \\ mod : & \begin{bmatrix} arg : \mathrm{Soa} & \\ val|nuc : & \begin{bmatrix} qfsoa : \begin{bmatrix} \textsc{accidental-soa} \\ soa\text{-}arg : \mathrm{Soa} \end{bmatrix} \\ restr : \langle\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 4: A restrictive adverbial and an operator adverbial. Restrictive adverbials (such as locatives and time adverbials) will generally be encoded as presented, where R_0 is a meta-variable that is instantiated by the restriction introduced by the adjunct. Operator adverbials (such as causatives) on the other hand introduce their own quantified state of affairs. Such adverbials generally are encoded as in the following example of the adverbial 'toevallig' (accidentally). Adverbials of the first type add a restriction to the semantics of the verb; adverbials of the second type introduce a new scope of modification.

$$\begin{bmatrix} \textsc{verbal} & \\ sc : & \left\langle \begin{bmatrix} \textsc{verbal} \\ sc : \langle \mathrm{E}, \mathrm{H} \rangle \\ lex : \text{kussen} \\ dir : \text{right} \end{bmatrix}, \begin{bmatrix} \textsc{adverbial} \\ lex : \text{vandaag} \\ dir : \text{left} \end{bmatrix}_{\mathrm{E}}, \begin{bmatrix} \textsc{noun} \\ lex : \text{bob} \\ dir : \text{left} \end{bmatrix}_{\mathrm{H}}, \begin{bmatrix} \textsc{noun} \\ lex : \text{arie} \\ dir : \text{left} \end{bmatrix} \right\rangle \\ lex : \text{wil} & \end{bmatrix}$$

$$\begin{bmatrix} \textsc{verbal} & \\ sc : & \left\langle \begin{bmatrix} \textsc{verbal} \\ sc : \langle \mathrm{H} \rangle \\ lex : \text{kussen} \\ dir : \text{right} \end{bmatrix}, \begin{bmatrix} \textsc{adverbial} \\ lex : \text{vandaag} \\ dir : \text{left} \end{bmatrix}, \begin{bmatrix} \textsc{noun} \\ lex : \text{bob} \\ dir : \text{left} \end{bmatrix}_{\mathrm{H}}, \begin{bmatrix} \textsc{noun} \\ lex : \text{arie} \\ dir : \text{left} \end{bmatrix} \right\rangle \\ lex : \text{wil} & \end{bmatrix}$$

Figure 5: The final instantiation of the modal for both the narrow- and the wide-scope reading of the sentence 'Arie Bob vandaag wil kussen'. In the narrow-scope reading the adverbial occurs both on the subcat list of the embedded verb and on the subcat list of the matrix verb — indicating that the embedded verb introduced the adjunct. In the wide-scope reading the adverb only occurs on the subcat list of the matrix verb.

We are now in a position to explain the observed ambiguity of adjuncts in verb-cluster constructions. Cf.:

(12) dat Arie Bob vandaag wil kussen
that Arie Bob today wants to-kiss

In the narrow-scope reading the adjunct is first added to the subcat list of 'kussen' and then passed on to the subcat list of the auxiliary verb. In the wide-scope reading the adjunct is added to the subcat list of the auxiliary verb. The final instantiations of the auxiliary 'wil' for both readings are given in figure 5.

2.2 Discussion

A further problem concerning the syntax of adjuncts is posed by the fact that adjuncts can take part in unbounded dependency constructions. Lexical treatments of the kind presented in (Pollard and Sag, in press, chapter 9) assume that a lexical rule is responsible for 'moving' an element from the subcat list to the slash list. Such an account predicts that adjuncts can not take part in such unbounded dependency constructions. In (Pollard and Sag, in press, chapter 9) a special rule is introduced to account for those cases where adjuncts do take part in UDCs. The treatment that we propose for adjuncts obviates the need for such an 'ad-hoc' rule.

Clearly many details concerning the syntax of adjuncts are left untouched here, such as the quite subtle restrictions in word-order possibilities of certain adjuncts with respect to arguments and with respect to other adjuncts. In the current framework linguistic insights concerning these issues could be expressed as constraints on the resulting subcategorization list (e.g. by means of LP-constraints).

It should also be stressed that treating adjuncts and arguments on a par on the level of subcategorization does not imply that observed differences in the behavior of adjuncts and arguments could not be handled in the proposed framework. For example the difference of adjuncts and arguments in the case of left dislocation in Dutch (exemplified in 13–16) can be treated by a lexical rule that operates on the subcat list before adjuncts are added.

(13) De voorstelling duurt een uur
The show takes an hour

(14) Een uur, dat duurt de voorstelling

(15) Arie en Bob wandelen een uur
Arie and Bob walk an hour

(16) * Een uur, dat wandelen Arie en Bob

3 Processing Lexical Rules

3.1 Lexical Rules as Constraints on Lexical Categories

Rather than formalizing the 'add-adjuncts' rule as a lexical rule we propose to use recursive constraints on lexical categories. Such lexical constraints are then processed using delayed evaluation techniques. [3]

Such an approach is more promising than an off-line approach that precomputes the effect of lexical rules by compilation of the lexicon, as it is unclear how recursive lexical rules can be treated in such an architecture (especially since some recursive rules can easily lead to an infinite number of lexical entries, e.g. the adjuncts rule).

Another alternative is to consider lexical rules as 'ordinary' unary rules. If this technique is applied for the lexical rules we have envisaged here, then (unary) derivations with unbounded length have to be considered.

If we formalize lexical rules as (complex) constraints on lexical categories then we are able to use delayed evaluation techniques for such constraints.

Assume that the 'underlying' feature structure of a verb is given by a definition of 'stem' (e.g. as the example of 'wil' above, or as the example of a simple transitive verb such as 'kussen' (to-kiss) in figure 6).

Such a feature-structure is not the actual category of the verb — rather this category is defined with complex constraints with respect to this base form. Here the constraint that adds adjuncts to the subcat list has our special attention, but there is also a constraint that adds a subject to the subcat list (as part of the inflection constraint for finite verbs) and a constraint that pushes an element from the subcat list to slash (to treat unbounded dependencies along the lines of chapter 9 of (Pollard and Sag, in press)), etc. Thus a lexical entry might be defined as in figure 7.

Lexical rules are regarded as (complex) constraints in this framework because it allows an implementation using delayed evaluation techniques from logic programming. The idea is that a certain constraint is only (partially) evaluated if 'enough' information is available to do so

[3] Refer to (Carpenter, 1991) for a proof of Turing equivalence of simple categorial grammar with recursive lexical rules.

$$\begin{bmatrix} \text{VERBAL} \\ sc : \langle \begin{bmatrix} \text{NOUN} \\ sem : \mathrm{A}_2 \end{bmatrix} \rangle \\ subj : \begin{bmatrix} \text{NOUN} \\ sem : \mathrm{A}_1 \end{bmatrix} \\ sem|nuc|qfsoa : \begin{bmatrix} \text{KISS-SOA} \\ kisser : \mathrm{A}_1 \\ kissed : \mathrm{A}_2 \end{bmatrix} \end{bmatrix}$$

Figure 6: Category for 'kussen' (to kiss)

lexical_entry(A) :-
stem(B), add_adj(B, C),
inflection(C, D), push_slash(D, A).

$$\text{inflection}(\begin{bmatrix} \text{VERBAL} \\ phon : \mathrm{P} \\ sc : \mathrm{Sc} \\ subj : \mathrm{Subj} \end{bmatrix}, \begin{bmatrix} \text{FINITE} \\ phon : \mathrm{P} \oplus \text{“t”} \\ sc : \mathrm{Sc} \cdot \langle \mathrm{Subj} \rangle \\ subj : \mathrm{Subj} \end{bmatrix}).$$

Figure 7: A lexical entry is defined with respect to a base form using complex constraints. Subject addition is a constraint associated with finite inflection.

successfully. As a relatively simple example we consider the constraint that is responsible for adding a subject as the last element on a subcat list of finite verbs. As a lexical rule we might define:

$$\begin{bmatrix} \text{FINITE} \\ subj : \mathrm{Subj} \\ sc : \mathrm{Sc} \end{bmatrix} \Longrightarrow \begin{bmatrix} sc : \mathrm{Sc} \cdot \langle \mathrm{Subj} \rangle \end{bmatrix}$$

If we use constraints the definition can be given as in figure 7, as part of the constraint associated with finite morphology. Note that the two approaches are not equivalent. If we use lexical rules then we have to make sure that the add-subject rule should be applied only once, and only for finite verbs. As a constraint we simply call the constraint once at the appropriate position.

The concatenation constraint (associated with the 'dot' notation) is defined as usual:

concat(⟨⟩, A, A).
concat(⟨B|C⟩, A, ⟨B|D⟩) :-
concat(C, A, D).

If this constraint applies on a category of which the subcat list is not yet fully specified (for example because we do not yet know how many adjuncts have been added to this list) then we cannot yet compute the resulting subcat list. The constraint can be successfully applied if either one of the subcat lists is instantiated: then we obtain a finite number of possible solutions to the constraint.

The relation *add_adj* recursively descends through a subcategorization list and at each position either adds or does not add an adjunct (of the appropriate type). Its definition is given in figure 8. Note that it is assumed in this definition that the scope of (operator-type) adverbials is given by the order in which they are put in in the subcategorization list, i.e. in the obliqueness order. [4]

[4] Cf. (Kasper, in preparation) for discussion of this point, also in relation with adjuncts that introduce quantifiers. Note that in our approach different possibilities can be defined.

$$\text{add\_adj}(\begin{bmatrix}\text{SIGN}\\ sc:\text{A}\\ sem:\text{B}\\ subj:\text{Subj}\end{bmatrix}, \begin{bmatrix}\text{SIGN}\\ sc:\text{J}\\ sem:\text{K}\\ subj:\text{Subj}\end{bmatrix})\texttt{:-}$$

$$\quad \text{add\_adj}(\text{A},\text{J},\text{B},\text{K}).$$

$$\text{add\_adj}(\langle\rangle,\langle\rangle,\text{A},\text{A}).$$
$$\text{add\_adj}(\langle\text{C}|\text{D}\rangle,\langle\text{C}|\text{E}\rangle,\text{A},\text{B})\texttt{:-}$$
$$\quad \text{add\_adj}(\text{D},\text{E},\text{A},\text{B}).$$
$$\text{add\_adj}(\text{A},\langle \begin{bmatrix}\text{ADVERBIAL}\\ mod:\begin{bmatrix}\text{MOD}\\ arg:\text{B}\\ val:\text{E}\end{bmatrix}\end{bmatrix} |\text{D}\rangle,\text{B},\text{C})\texttt{:-}$$
$$\quad \text{add\_adj}(\text{A},\text{D},\text{E},\text{C}).$$

Figure 8: Definite clause specification of 'add_adj' constraint.

3.2 Delayed evaluation

For our current purposes, the co-routining facilities offered by Sicstus Prolog are powerful enough to implement a delayed evaluation strategy for the cases discussed above. For each constraint we declare the conditions for evaluating a constraint of that type by means of a `block` declaration. For example the *concat* constraint is associated with a declaration:

`:- block` $concat(-,?,-)$.

This declaration says that evaluation of a call to *concat* should be delayed if both the first and third arguments are currently variable (uninstantiated, of type TOP). It is clear from the definition of *concat* that if these arguments are instantiated then we can evaluate the constraint in a top-down manner without risking non-termination. E.g. the goal $concat(\langle \text{A},\text{B}\rangle,\text{C},\text{D})$ succeeds by instantiating D as the list $\langle \text{A},\text{B}|\text{C}\rangle$.

Note that block declarations apply recursively. If the third argument to a call to *concat* is instantiated as a list with a variable tail, then the evaluation of the recursive application of that goal might be blocked; e.g. evaluation of the goal $concat(\text{A},\langle\text{Sj}\rangle,\langle\text{B}|\text{C}\rangle)$ succeeds either with both A and C instantiated as the empty list and by unifying Sj and B, or with A instantiated as the list $\langle\text{B}|\text{D}\rangle$ for which the constraint $concat(\text{D},\langle\text{Sj}\rangle,\text{C})$ has to be satisfied. Similarly, for each of the other constraints we declare the conditions under which the constraint can be evaluated. For the *add_adj* constraint we define:

`:- block` $add\_adj(?,-,?,?)$.

One may wonder whether in such an architecture enough information will ever become available to allow the evaluation of any of the constraints. In general such a problem may surface: the parser then finishes a derivation with a large collection of constraints that it is not allowed to evaluate — and hence it is not clear whether the sentence associated with that derivation is in fact grammatical (as there may be no solutions to these constraints).

The strategy we have used successfully so-far is to use the structure hypothesized by the parser as a 'generator' of information. For example, given that the parser hypothesizes the application of rules, and hence of certain instantiations of the subcat list of the (lexical) head of such rules, this provides information on the subcat-list of lexical categories. Keeping in mind the definition of a lexical entry as in figure 7 we then are able to evaluate each of the constraints on the value of the subcat list in turn, starting with the *push_slash* constraint, up through the *inflection* and *add_adj* constraints. Thus rather than using the constraints as 'builders' of subcat-lists the constraints are evaluated by checking whether a subcat-list hypothesized by the parser can be related to a subcat-list provided by a verb-stem. In other words, the flow of information in the definition of *lexical_entry* is not as the order of constraints might suggest (from top to bottom) but rather the other way around (from bottom to top).

4 Final remarks

We illustrated that recursive lexical constraints might be useful from a linguistic perspective. If lexical rules are formalized as complex constraints on lexical categories then methods from logic programming can be used to implement such constraints.

Note that complex constraints and delayed evaluation techniques are also useful in other areas of linguistic desciption. For example we used the same methods to define and process HPSG's FOOT FEATURE PRINCIPLE. The method may also be applied to implement HPSG's binding theory.

As a testcase we improved upon the HPSG analysis of (Germanic) verb clusters and adjuncts by treating adjuncts as categories that are on the subcat list by virtue of a complex constraint. The fragment that has been implemented with the methods described is much larger than the discussion in the previous sections suggest, but includes treatments of *extraposition, ipp, modal inversion, participium inversion, the third construction, partial-vp topicalisation, particle verbs, verb-second, subject raising, subject control, raising-to-object, object control and clitic climbing* in Dutch.

References

Carpenter, Bob. 1991. The generative power of categorial grammars and head-driven phrase structure grammars with lexical rules. *Computational Linguistics*, 17(3):301–313.

Hinrichs, Erhard and Tsuneko Nakazawa. 1989. Flipped out: AUX in german. In *Papers from the 25th Annual Regional Meeting of the Chicago Linguistic Society*. Chicago Linguistics Society, Chicago, pages 187–202.

Kasper, Robert. in preparation. Adjuncts in the mittelfeld. In John Nerbonne, Klaus Netter, and Carl Pollard, editors, *German Grammar in HPSG*, Lecture Note Series. CSLI, Stanford.

Miller, Philip. 1992. *Clitics and Constituents in Phrase Structure Grammar*. New York: Garland.

Moortgat, Michael. 1988. *Categorial Investigations*. Ph.D. thesis, University of Amsterdam.

Pollard, Carl and Ivan Sag. 1987. *Information Based Syntax and Semantics, Volume 1*. Center for the Study of Language and Information Stanford.

Pollard, Carl and Ivan Sag. in press. *Head-driven Phrase Structure Grammar*. Center for the Study of Language and Information Stanford.