Valence Alternation without Lexical Rules

Gosse Bouma*

Abstract

Valence changing lexical rules are a problematic component of constraint-based grammar formalisms. Lexical rules of this type are procedural, require defaults, and may easily lead to spurious ambiguity. Relational constraints can be used to eliminate such rules. The relational approach does not require defaults, is declarative, avoids spurious ambiguity, and can be an integrated part of a hierarchically structured lexicon. This is illustrated below for the complement extraction and adjunct introduction lexical rules of HPSG. We argue that, apart from the technical benefits mentioned above, the relational approach is linguistically superior, in that it offers a uniform account of complement and adjunct extraction. Furthermore, it eliminates the spurious ambiguity that may arise in grammars which include complement inheritance verbs as well as a lexicalist account of complement extraction.

Introduction

Recent work in HPSG has argued for lexicalist approaches to complement extraction (Sag 1995), adjunct selection (Miller 1992; Iida, Manning, O'Neill, and Sag 1994; Manning, Sag, and Iida 1996), and clitic climbing (Sag and Miller, to appear). Lexicalist accounts treat these phenomena as valence variation. That is, complement extraction requires that each head selecting for an extractable complement C has a counterpart which does not select for C but instead includes C in its SLASH-set. Similarly, lexicalist adjunct selection requires that heads may include (an arbitrary number of) adjuncts on their COMPS-list. Lexicalist accounts of clitic climbing, finally, require not only that lexical heads may realize some of their complements as phonological clitics, but also that these heads have a COMPS-list which is the `append` of the list of elements the head subcategorizes for and the COMPS-list of one of these elements. That is, verbs allowing for clitic climbing must be *complement inheritance* verbs. Note that both the phonological realization of complements as clitics and complement inheritance lead to valence alternations.

The proposals cited above all rely on lexical rules to account for certain systematic alternations in lexical entries. The central role of lexical rules is remarkable, given the fact that lexical rules are often seen as more or less *ad hoc*, procedural, extensions of the formalism, whose formal status is far from resolved.

At the same time, it has been customary in HPSG to employ relational, recursive, constraints. That is, in accounts of phenomena such as extraction, complement

*Rijksuniversiteit Groningen, vakgroep Alfa-informatica

inheritance, quantifier scoping, and word order, the values of list and set-valued features are routinely defined using relations such as `member`, `delete`, `append`, and `(sequence) union`. Lexical rules in particular, are often defined using such relations.

Finally, the treatment of valence in recent versions of HPSG is more subtle than in the versions presented in Pollard and Sag (1987) and Pollard and Sag (1994, chapters 1-8). Following Borsley (1989), it is now customary to distinguish subjects from other complements by means of the two valence features SUBJ and COMPS (replacing the single SUBCAT feature used before). Furthermore, valence is distinguished from *argument structure*, represented by the feature ARG-ST. Argument structure contains the list of elements which a lexical sign selects for and is the level of representation to which the binding principles apply. While in the canonical case ARG-ST will correspond to the `append` of SUBJ and COMPS, this is by no means always true. In Manning and Sag (1995) it is observed that phenomena such as passive, 'pro-drop', and syntactic ergativity in a number of languages can be seen as evidence for several non-canonical relationships between valence and argument structure, providing evidence for a level of representation independent of valence. Note also that complement inheritance verbs will typically contain (inherited) elements on COMPS that do not correspond to arguments of that verb. Lexicalized extraction, finally, implies that some (non-subject) elements on ARG-ST will not be present on COMPS, but are included in SLASH instead.

In this paper, it is argued that the distinction between valence and argument structure allows valence changing lexical rules to be eliminated. Valence alternations are captured instead by general, possibly recursive, constraints defining the mapping between argument structure and valence. We demonstrate this in some detail for complement extraction and adjunct introduction. Other valence changing lexical rules (such as) can in principle be replaced by constraints in a similar fashion.[1]

Approaches to valence variation using relational constraints have been proposed by, among others, Kathol (1994) and Frank (1994). The current proposal, however, allows recursive constraints, and thus it can account for complement extraction (requiring arbitrary elements on ARG-ST to be realized as *gaps*) and adjunct introduction (requiring the insertion of an arbitrary number of adjuncts on ARG-ST (and COMPS)). Furthermore, the constraints proposed below do not require the introduction of additional features. Instead, all constraints apply to independently motivated features, leading to a tight integration of the constraint system with the overall architecture of HPSG.

The connection between lexical rules and relational constraints was first noted in van Noord and Bouma (1994). By viewing lexical rules as relational constraints, delayed evaluation techniques can be used to solve the computational problems posed by recursive lexical rules. However, the constraints in van Noord and Bouma (1994) hold between full-blown lexical entries (i.e. signs), whereas below we use

[1] In Sag and Miller (to appear), for instance, an account of French clitization is presented which is directly compatible with (and inspired by) the approach outlined below in that it defines the realization of certain elements on ARG-ST as clitics by means of a constraint on the mapping between ARG-ST and COMPS, instead of by means of a lexical rule, as in previous proposals.

constraints to relate only specific features within a sign. Since all constraints apply to the same sign conjunctively, the issue of rule-ordering, which was solved in van Noord and Bouma (1994) by hard-wiring the order of rule application into the constraints (see Meurers and Minnen (1995) for an alternative approach), disappears. Also, the need for default sharing of information between input and output of a lexical rule disappears.

Below, we present an example lexicon fragment, in which both lexical inheritance and lexical rules are used. We point out a number of problematic aspects of these rules in a constraint-based setting. In section 2, we redefine the fragment by replacing lexical rules with relational constraints. We demonstrate that the constraint-based fragment naturally leads to an account of complement extraction which subsumes the possibility of adjunct extraction. In section 3, we argue that the kind of spurious ambiguity noted in Hinrichs and Nakazawa (1996) does not arise in our proposal.

1 A lexicon fragment with lexical rules

We present a lexicon fragment for verbs which uses inheritance, constraints, and lexical rules. We point out various problematic aspects of this set-up.

The basic lexicon

A definite clause specification for the basic lexical entries of a small lexicon fragment is given in fig. 1. The unary predicate `basic-entry` defines the set of basic lexical entries in the language. A basic entry is of type *word*, and can be a major category (i.e. *v, n, etc.*) entry satisfying the `slash-amalgamation` constraint (introduced below). A verbal major category must satisfy `verbal-subcat` and `map-args`. The first defines the various verbal subcategorization types, whereas the latter defines the mapping between argument structure and valence.

Following Manning and Sag (1995), we assume that different verbal subcategorization types differ only in their argument structure, and that the values of the valence features are defined by means of a general *mapping* constraint. This is the task of the relational constraint `map-args`. Canonically, the first element on ARG-ST is the subject, while the rest is equal to COMPS ('|' connects the head and tail of a list). (Alternative definitions are considered below.). By combining the definitions of `verbal-subcat`, `verbal-lex`, and `map-args`, we can for instance derive the following fact:

$$(1)\ \texttt{major}(\begin{bmatrix} \text{PHON} & \textit{hates} \\ \text{HEAD} & v \\ \text{ARG-ST} & \langle \boxed{1}\ \text{NP}_i, \boxed{2}\ \text{NP}_j \rangle \\ \text{SUBJ} & \langle \boxed{1} \rangle \\ \text{COMPS} & \langle \boxed{2} \rangle \\ \text{CONT} & \textit{hate}'(i,j) \end{bmatrix})$$

$$\texttt{basic-entry}\left(\boxed{1}\begin{bmatrix} \textit{word} & \\ \text{ARG-ST} & \boxed{2} \\ \text{SLASH} & \boxed{3} \end{bmatrix}\right) \leftarrow$$

$$\qquad \texttt{major}(\boxed{1}) \wedge \texttt{slash-amalgamation}(\boxed{2}, \boxed{3})$$

$$\texttt{major}\left(\boxed{1}\begin{bmatrix} \text{HEAD} & v \\ \text{ARG-ST} & \boxed{2} \\ \text{SUBJ} & \boxed{3} \\ \text{COMPS} & \boxed{4} \end{bmatrix}\right) \leftarrow$$

$$\qquad \texttt{verbal-subcat}(\boxed{1}) \wedge \texttt{map-args}(\boxed{2}, \boxed{3}, \boxed{4})$$

$$\texttt{verbal-subcat}\left(\begin{bmatrix} \text{PHON} & \boxed{1} \\ \text{ARG-ST} & \langle\ \text{NP}_i\ \rangle \\ \text{CONT} & \boxed{2}(i) \end{bmatrix}\right) \leftarrow$$

$$\qquad \texttt{verbal-lex(intrans,}\boxed{1}\texttt{,}\boxed{2}\texttt{)}$$

$$\texttt{verbal-subcat}\left(\begin{bmatrix} \text{PHON} & \boxed{1} \\ \text{ARG-ST} & \langle\ \text{NP}_i, \text{NP}_j\ \rangle \\ \text{CONT} & \boxed{2}(i,j) \end{bmatrix}\right) \leftarrow$$

$$\qquad \texttt{verbal-lex(trans,}\boxed{1}\texttt{,}\boxed{2}\texttt{)}$$

$$\texttt{verbal-lex}(\texttt{instrans}, \textit{sleeps}, \textit{sleep}')$$

$$\texttt{verbal-lex}(\texttt{trans}, \textit{hates}, \textit{hate}')$$

$$\texttt{map-args}(\langle\ \boxed{1}\ \mid\ \boxed{2}\ \rangle, \langle\ \boxed{1}\ \rangle, \boxed{2}\ )$$

$$\texttt{slash-amalgamation}(\langle\ \rangle, \emptyset)$$

$$\texttt{slash-amalgamation}(\langle\ [\ \text{SLASH}\ \boxed{1}\ ]\ \mid\ \boxed{2}\ \rangle, \boxed{1} \uplus \boxed{3}) \leftarrow$$

$$\qquad \texttt{slash-amalgamation}(\boxed{2}, \boxed{3})$$

Figure 1: A fragment of the basic lexicon

The two main features of the lexicalist approach to extraction presented in Sag (1995) is the elimination of the NONLOCAL FEATURE PRINCIPLE in favour of a lexical *slash amalgamation* constraint and the elimination of traces in favour of a lexical complement extraction rule. Slash amalgamation requires that the SLASH-value of a basic lexical entry is the set-union of the SLASH-values of its arguments. The `slash-amalagamation` constraint implements this by recursively traversing the list of elements on ARG-ST, and unioning all the SLASH values: ($\uplus$ denotes (non-vacuous) set-union). Slash amalgamation makes the NONLOCAL FEATURE PRINCIPLE superfluous, as SLASH can simply be shared between head and mother in phrases without a filler daughter, while SLASH is subject to rule-specific constraints in *head-filler* phrases. An example of slash amalgamation at work is given after we have introduced the lexical rule for extraction.

The basic lexicon incorporates the following notion of *lexical inheritance*: A basic entry has various major category entries as its subclasses. All of these subclasses must satisfy `slash-amalgamation`. Similarly, the verbal major category class has various verbal subcategorization types as subclass. All of these must satisfy `map-args`. Thus, the unary predicates in general define subclasses of the general class `basic-entry`, whereas the other predicates define constraints which must hold for the class in whose antecedent the predicate appears.

Adding lexical rules

In fig. 2, we define two lexical rules. A lexical rule defines a relationship between an 'input' and 'output' lexical entry. Therefore, lexical rules can be added to the fragment as instances of the relation `lexical-rule`(*In*,*Out*). Furthermore, the set of lexical entries (basic or derived) is now defined by the relation `entry`.

$\texttt{entry}(\boxed{1}) \leftarrow$

$\quad \texttt{basic-entry}(\boxed{1})$

$\texttt{entry}(\boxed{1}) \leftarrow$

$\quad \texttt{entry}(\boxed{0}) \wedge \texttt{lexical-rule}(\boxed{0},\boxed{1})$

%% complement extraction lexical rule (celr)

$$\texttt{lexical-rule}\left(\begin{bmatrix}\text{COMPS} & \boxed{1} \bigcirc \left\langle \begin{bmatrix}\text{LOC} & \boxed{2} \\ \text{SLASH} & \{\ \boxed{2}\ \}\end{bmatrix} \right\rangle\end{bmatrix}, \begin{bmatrix}\text{COMPS} & \boxed{1}\end{bmatrix}\right)$$

%% adjuncts lexical rule

$$\texttt{lexical-rule}\left(\begin{bmatrix}\text{ARG-ST} & \boxed{1} \\ \text{CONT} & \boxed{2}\end{bmatrix}, \begin{bmatrix}\text{ARG-ST} & \boxed{1} \oplus \langle\ \text{ADV}\ \rangle \\ \text{CONT} & adv\prime(\ \boxed{2}\ )\end{bmatrix}\right)$$

Figure 2: Adding lexical rules

The complement extraction lexical rule (CELR) is adopted from Sag (1995), and identifies an element on COMPS as a *gap* (i.e. subtype of *synsem* which satisfies the constraint that its SLASH-value is a singleton set, the only element of which is reentrant with LOC). The *gap* is absent in the output of the rule ($\bigcirc$ denotes sequence union). The interaction of this rule with slash amalgamation implies that the SLASH-value of the deleted element will be included in SLASH of the input (and output) sign. As each complement is also present on ARG-ST, and the SLASH-value of the sign itself is the union of the SLASH-value of its members, the instantiation of SLASH on one of these members will have a direct effect on SLASH. Furthermore, as it is assumed that information is shared by default between input and output, the instantiated SLASH value will be present on the output of the rule as well. Note, however, that the present formulation does not account for default sharing of information.

Assuming the latter problem can be solved, the CELR allows the derivation of the following lexical entry, in which the object has been extracted:

$$(2)\quad \texttt{entry}(\begin{bmatrix} \text{PHON} & \textit{hates} \\ \text{ARG-ST} & \left\langle \boxed{1}\ \text{NP}_i[\text{SLASH}\ \boxed{3}],\ \begin{bmatrix} \text{LOC} & \boxed{2}\ \text{NP}_j \\ \text{SLASH} & \{\ \boxed{2}\ \} \end{bmatrix} \right\rangle \\ \text{SUBJ} & \langle\ \boxed{1}\ \rangle \\ \text{COMPS} & \langle\ \rangle \\ \text{SLASH} & \boxed{3} \uplus \{\ \boxed{2}\ \} \end{bmatrix})$$

Together with slash amalgamation, and the assumption that SLASH is a head feature in head-valence phrases, while it gets 'bound' in head-filler phrases, this allows for the derivations of the example in fig. 3.

The second lexical rule lexically introduces adjuncts as complements. Several versions of such a rule have been presented (Miller 1992; van Noord and Bouma 1994; Manning, Sag, and Iida 1996). Here, we will assume, following Manning, Sag, and Iida (1996), that adjuncts are added to ARG-ST for reasons of binding and (adjunct) extraction ($\oplus$ denotes append).

Again, this rule as given is incomplete. However, an appeal to default matching cannot give the correct results in this case. Note that the newly introduced adjunct should be added to COMPS as well. This means that the value of COMPS on input and output must differ, in spite of the fact that the rule does not mention them. Intuitively, the correct value for COMPS should follow from the `map-args` constraint. It is unclear, however, how that constraint could be made to apply at this point. For one thing, the interaction with complement extraction (which creates exceptions to the canonical mapping relation) appears to be highly problematic. That is, a lexical entry derived by means of complement extraction contains an element on ARG-ST which is not realized on COMPS (i.e. the derived entry for *kiss* above). Adding an adjunct to such an entry and reapplying `map-args` to the result would reintroduce the extracted complement.

A similar difficulty arises in trying to account for adjunct extraction. The CELR relies on the fact that `slash-amalgamation` takes into account all elements on ARG-ST. However, the adjuncts rule introduces new elements on ARG-ST. This

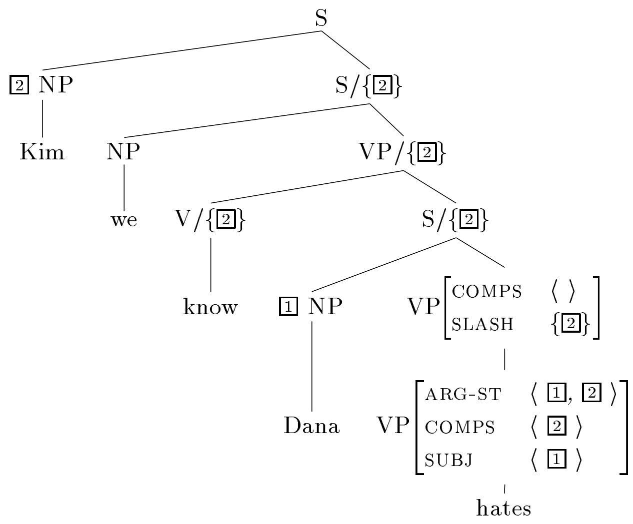

Figure 3: Kim, we know Dana hates

implies that extracting an adjunct by means of the CELR only has the intended effect if `slash-amalgamation` is used to 'recompute' the value of SLASH on the output of the rule.

Problems for lexical rules

We conclude this section with an overview of problematic aspects of lexical rules in a constraint-based setting.

Default sharing between input and output. Lexical rules typically affect only a small part of the information in a lexical entry. To account for the similarity between input and output, some kind of default sharing of information is required. Default unification as defined in Bouma (1992), Carpenter (1992) or Lascarides, Briscoe, Asher, and Copestake (1996) either is not applicable to the typed constraint language presupposed by HPSG or to the problem of default sharing in lexical rules. Therefore, Meurers (1995) proposes a special-purpose default mechanism for lexical rules. Even if this problem can be solved, it is still the case that lexical rules are the only component of HPSG where nonmonotonicity comes into play.

Interaction with Inheritance. The adjuncts lexical rule illustrates clearly that in some cases one wants to use inheritance of constraints to fill in missing information in the output, instead of default sharing. The discussion of lexical rules in Pollard and Sag (1987, chapter 8) also makes this assumption. No detailed proposals for such an interpretation of lexical rules exist, however. The conflict between these two interpretations of lexical rules also seems to have gone unnoticed in the literature.

Spurious ambiguity. The complement extraction lexical rule removes an element from COMPS. If this rule is used to delete two elements, say C_1 and C_2, one could either remove C_1 first, or C_2. The distinction is irrelevant for the outcome, however. Similarly, complement extraction removes an element, while adjunct introduction adds an element. Again, both orders are possible, but in general (the exception being cases of adjunct extraction) this will lead to the same result. To eliminate this kind of redundancy, one may have to introduce external rule ordering, reformulate the rules so that they no longer need to apply recursively (van Noord and Bouma 1994), or add finite state control devices (Meurers and Minnen 1995).

Subsumption. Hinrichs and Nakazawa (1996) have argued that lexical rules should only be applied to lexical entries that are subsumed by the input conditions of the rule. However, not all lexical rules can be interpreted this way. Also, checking for subsumption appears to be incompatible with certain processing strategies. We return to this issue in section 3.

2 A constraint-based alternative

A radical solution for the problems just mentioned is to eliminate lexical rules and to account for valence variation by means of (recursive) constraints only. On the one hand, the elimination of lexical rules is a substantial simplification of the formalism. On the other hand, using recursive constraints for valence alternations is not a complication of the formalism, as recursive constraints are used in various other components of HPSG already. Note also that lexical rules in particular tend to be defined in terms of recursive constraints. That is, arguments that a system with lexical rules allows fewer or simpler constraints than a system without lexical rules can be rejected easily.

The fragment presented in section 1 defines the relationship between argument structure and valence by means of a mapping constraint. Valence changing lexical rules, such as the complement extraction lexical rule, typically derive lexical entries which do not obey the 'canonical mapping'. A more principled approach to valence alternations, therefore, is to take these lexical entries not as exceptions, derived by means of a rule, but to redefine the mapping between argument structure and valence, so as to allow for the 'exceptional' cases as well.

The definitions in fig. 4 provide a reformulation of the definition of major verbal category lexical entries presented in fig. 1. The CELR and adjuncts lexical rules are made superfluous by a reformulation of `map-args` and the introduction of an `adjuncts` constraint on verbal lexical entries.

Complement extraction

The `map-args` constraint relates argument structure to the valence features SUBJ and COMPS, as before. The new `map-non-subj-args` constraint ensures that non-subject arguments are either realized as complements, or as *gaps*. The range of lexical entries satisfying `map-args` as defined in fig. 4 therefore corresponds exactly to what can be derived by means of the CELR, making the latter spurious.

$$\texttt{major}(\begin{bmatrix} \text{PHON} & \boxed{0} \\ \text{HEAD} & v \\ \text{ARG-ST} & \boxed{1} \oplus \boxed{5} \\ \text{SUBJ} & \boxed{2} \\ \text{COMPS} & \boxed{3} \\ \text{CONT} & \boxed{4} \end{bmatrix}) \leftarrow$$

$$\texttt{verbal-subcat}(\begin{bmatrix} \text{PHON} & \boxed{0} \\ \text{ARG-ST} & \boxed{1} \\ \text{CONT} & \boxed{6} \end{bmatrix}) \wedge$$

$$\texttt{adjuncts}(\boxed{5}, \boxed{6}, \boxed{4}) \wedge$$

$$\texttt{map-args}(\boxed{1} \oplus \boxed{5}, \boxed{2}, \boxed{3})$$

%% map-args(Arg-st,Subj,Comps)

$$\texttt{map-args}(\langle \boxed{1} | \boxed{2} \rangle, \langle \boxed{1} \rangle, \boxed{3}) \leftarrow$$

$$\texttt{map-non-subj-args}(\boxed{2}, \boxed{3})$$

%% map-non-subj-args(Arg-st,Comps)

$$\texttt{map-non-subj-args}(\langle\,\rangle, \langle\,\rangle)$$

$$\texttt{map-non-subj-args}(\langle \boxed{1} \mid \boxed{2} \rangle, \langle \boxed{1} \mid \boxed{3} \rangle) \leftarrow$$

$$\texttt{map-non-subj-args}(\boxed{2}, \boxed{3})$$

$$\texttt{map-non-subj-args}(\left\langle \begin{bmatrix} \text{LOC} & \boxed{1} \\ \text{SLASH} & \{ \boxed{1} \} \end{bmatrix} \mid \boxed{2} \right\rangle, \boxed{3}) \leftarrow$$

$$\texttt{map-non-subj-args}(\boxed{2}, \boxed{3})$$

%% adjuncts(Arg-st, Cont, Cont)

$$\texttt{adjuncts}(\langle\,\rangle, \boxed{1}, \boxed{1})$$

$$\texttt{adjuncts}(\langle \text{ADV} \mid \boxed{1} \rangle, \boxed{2}, \boxed{3}) \leftarrow$$

$$\texttt{adjuncts}(\boxed{1}, adv'(\boxed{2}), \boxed{3})$$

Figure 4: A fragment without lexical rules

For instance, assume that the following can be derived by resolving `major` with `verbal-subcat` and `adjuncts` :

$$(3)\ \ \texttt{major}(\begin{bmatrix} \text{PHON} & \textit{hates} \\ \text{ARG-ST} & \boxed{1}\ \langle\ \text{NP}_i,\ \text{NP}_j\ \rangle \\ \text{SUBJ} & \boxed{2} \\ \text{COMPS} & \boxed{3} \end{bmatrix}) \leftarrow \texttt{map-args}(\boxed{1}, \boxed{2}, \boxed{3})$$

This can be resolved with `map-args` to give rise to exactly the following two results:

$$(4)\ \ \text{a. } \texttt{major}(\begin{bmatrix} \text{PHON} & \textit{hates} \\ \text{ARG-ST} & \langle\ \boxed{1}\ \text{NP},\ \boxed{2}\ \text{NP}\ \rangle \\ \text{SUBJ} & \langle\ \boxed{1}\ \rangle \\ \text{COMPS} & \langle\ \boxed{2}\ \rangle \end{bmatrix})$$

$$\text{b. } \texttt{major}(\begin{bmatrix} \text{PHON} & \textit{hates} \\ \text{ARG-ST} & \left\langle \boxed{1}\ \text{NP}, \begin{bmatrix} \text{LOC} & \boxed{2}\ \text{NP} \\ \text{SLASH} & \{\ \boxed{2}\ \} \end{bmatrix} \right\rangle \\ \text{SUBJ} & \langle\ \boxed{1}\ \rangle \\ \text{COMPS} & \langle\ \ \rangle \end{bmatrix})$$

Note that `map-args` imposes a *constraint on* lexical entries, and does not define a *relation between* lexical entries. Since all lexical entries, or at least all verbs, must satisfy `map-args`, and since there is no distinction between a 'basic' and a 'derived' lexical entry, the issue of default sharing of information disappears.

A second advantage is that `map-non-subj-args` recursively traverses ARG-ST and (nondeterministically) 'decides' for each element whether it is to be realized as complement or as gap. Consequently, the spurious ambiguity that was observed for the CELR in case multiple complements had to be extracted, does not arise in the constraint-based approach.

Adding adjuncts

In section 1, we assumed that the argument structure of a verb is fully determined by its subcategorization type. The adjuncts lexical rule, however, is incompatible with this assumption, as it appends elements to ARG-ST which the verb does not subcategorize for.

The conflict can be resolved by assuming that argument structure is the `append` of two lists ($\boxed{1} \oplus \boxed{5}$ in the definition of `verb` in fig. 4), where the value of the first is determined by the verbal subcategorization type, and the value of the second is determined by the `adjuncts` constraint. A similar modification is necessary to account for semantics (CONT). As adjunct semantics takes the basic verbal semantics as argument, the semantics of a verb is no longer directly determined by choosing a particular instance of `verbal-subcat`. Instead, `verbal-subcat` supplies a basic semantic value which is taken as argument in the `adjuncts` constraint. The latter actually determines the CONT value of `verb`.

The effect of these modifications can be illustrated as follows. Assume that `major` resolves with `verbal-subcat` to give rise to the following:

$$(5)\ \texttt{major}(\left[\begin{array}{ll} \textsc{phon} & \textit{hates} \\ \textsc{arg-st} & \boxed{1}\ \langle\ \mathrm{NP}_i,\ \mathrm{NP}_j\ \rangle \oplus \boxed{5} \\ \textsc{subj} & \boxed{2} \\ \textsc{comps} & \boxed{3} \\ \textsc{cont} & \boxed{4} \end{array}\right]) \leftarrow$$

$$\texttt{adjuncts}(\boxed{5}, \textit{hate}'(i,j), \boxed{4}) \wedge \texttt{map-args}(\boxed{1}\oplus\boxed{5},\ \boxed{2},\ \boxed{3})$$

Resolution with `adjuncts`, among others, gives:

$$(6)\ \texttt{major}(\left[\begin{array}{ll} \textsc{phon} & \textit{hates} \\ \textsc{arg-st} & \boxed{1}\ \langle\ \mathrm{NP}_i,\ \mathrm{NP}_j,\ \mathrm{ADV}\ \rangle \\ \textsc{subj} & \boxed{2} \\ \textsc{comps} & \boxed{3} \\ \textsc{cont} & \textit{adv}'(\textit{hate}'(i,j)) \end{array}\right]) \leftarrow \texttt{map-args}(\boxed{1},\ \boxed{2},\ \boxed{3})$$

which in turn can be resolved against `map-args` to give:

$$(7)\ \texttt{major}(\left[\begin{array}{ll} \textsc{phon} & \textit{hates} \\ \textsc{arg-st} & \langle\ \boxed{1}\ \mathrm{NP}_i,\ \boxed{2}\ \mathrm{NP}_j,\ \boxed{3}\ \mathrm{ADV}\ \rangle \\ \textsc{subj} & \langle\ \boxed{1}\ \rangle \\ \textsc{comps} & \langle\ \boxed{2},\ \boxed{3}\ \rangle \\ \textsc{cont} & \textit{adv}'(\textit{hate}'(i,j)) \end{array}\right])$$

The newly introduced `adjuncts` constraint has exactly the same effect as the corresponding lexical rule. Since the constraint is integrated with the lexical hierarchy, however, the mapping between argument structure and valence is automatically accounted for.

Adjunct extraction

Since the possibility of adjuncts on ARG-ST is now taken into account in the definition of verbal lexical entries (i.e. the definition of `major` as given in fig. 4), `slash-amalgamation` will automatically apply to adjuncts on ARG-ST as well. Furthermore, the mapping between argument structure and valence, defined by `map-args`, will also take adjuncts into account. As `slash-amalgamation` and `map-args` are the two constraints responsible for complement extraction, the possibility of adjunct extraction is now just a special case of complement extraction.

For instance, an alternative solution for (6) is:

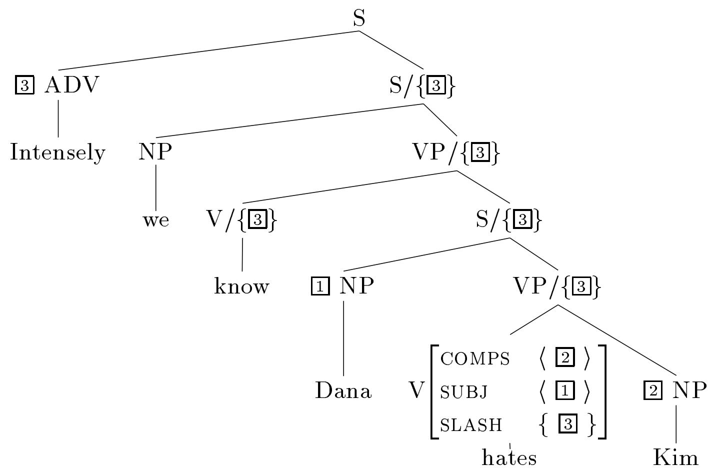

Figure 5: Intensely, we know Dana hates Kim

$$(8)\ \texttt{verb}\left(\begin{bmatrix} \text{PHON} & \textit{hates} \\ \text{ARG-ST} & \left\langle \boxed{1}\ \text{NP}_i,\ \boxed{2}\ \text{NP}_j,\ \begin{bmatrix} \text{LOC} & \boxed{3}\ \text{ADV} \\ \text{SLASH} & \{\ \boxed{3}\ \} \end{bmatrix} \right\rangle \\ \text{SUBJ} & \langle\ \boxed{1}\ \rangle \\ \text{COMPS} & \langle\ \boxed{2}, \boxed{3}\ \rangle \\ \text{CONT} & adv'(hate'(i,j)) \end{bmatrix}\right)$$

This allows us to derive the entry in (9) for *hates*, where `slash-amalgamation` has applied. An example involving this entry is given in fig. 5.

$$(9)\ \texttt{entry}\left(\begin{bmatrix} \text{PHON} & \textit{hates} \\ \text{ARG-ST} & \boxed{0}\left\langle \boxed{1}\ \text{NP}_i[\text{SL}\ \boxed{4}],\ \boxed{2}\ \text{NP}_j[\text{SL}\ \boxed{5}],\ \begin{bmatrix} \text{LOC} & \boxed{3}\ \text{ADV} \\ \text{SL} & \{\ \boxed{3}\ \} \end{bmatrix} \right\rangle \\ \text{SUBJ} & \langle\ \boxed{1}\ \rangle \\ \text{COMPS} & \langle\ \boxed{2}\ \rangle \\ \text{CONT} & adv'(hate'(i,j)) \\ \text{SLASH} & \boxed{4} \uplus \boxed{5} \uplus \{\ \boxed{3}\ \} \end{bmatrix}_{word}\right)$$

Constraints declaratively and monotonically define the space of possible lexical entries, whereas lexical entries do this procedurally and nonmonotonically. Therefore, constraints can be integrated into a hierarchical lexicon definition in a way that is difficult or impossible for a system using lexical rules. Furthermore, since the system is declarative, procedural issues such as rule ordering and spurious ambiguity do not arise. Since constraints relate specific features, and not (complete) lexical entries, default sharing of information also is no longer necessary.

There are also linguistic benefits. A grammar avoiding spurious ambiguity is linguistically preferable over a system which does allow spurious derivations. Also, as shown above, the constraint-based approach can account for the possibility of adjunct extraction in a way that does not require any additional rules or mechanisms.

3 Complement Inheritance and Extraction

In this section, we argue that the constraint-based approach also offers a solution for the spurious ambiguity problem observed in Hinrichs and Nakazawa (1996).

Hinrichs and Nakazawa (1994) have argued that German modal and auxiliary verbs are complement inheritance verbs, i.e. they subcategorize for a possibly unsaturated lexical verbal complement, and include the complements of this verb in their own COMPS list. That is, a modal verb such as German *können* (*to be able to*) must be associated with the feature structure in (10). In Hinrichs and Nakazawa (1996) it is argued that a combination of complement inheritance and an approach to complement extraction based on lexical rules leads to spurious ambiguity in sentences containing modal or auxiliary verbs, as an inherited complement may be extracted not only from the COMPS-list of the verb which subcategorizes for it, but also from the COMPS-list of each of the verbs inheriting this complement. This is illustrated in (11).

(10) $$\left[\begin{array}{ll} \textsc{phon} & \textit{können} \\ \textsc{arg-st} & \langle\ \boxed{1}\ \textsc{np}_i,\ \boxed{2}\ \textsc{v}_j[\textsc{comps}\ \boxed{3}\ ]\ \rangle \\ \textsc{subj} & \langle\ \boxed{1}\ \rangle \\ \textsc{comps} & \boxed{3} \oplus \langle\ \boxed{2}\ \rangle \\ \textsc{cont} & \textit{be-able}'(i,j) \end{array}\right])$$

(11) $\boxed{1}$ NP $\quad$ $\boxed{2}$ $\mathrm{V}\Big[\textsc{comps}\ \langle\ \boxed{1}\ \rangle\Big]$ $\quad$ $\mathrm{V}\Big[\textsc{comps}\ \langle\ \boxed{1}, \boxed{2}\ \rangle\Big]$

Welches Buch wird Peter kaufen können
`which book will Peter buy be-able`
Which book will Peter be able to buy

The extracted element *welches Buch* appears on the COMPS list of two verbs, and thus a complement extraction lexical rule could apply to either *kaufen* or *können.*

The solution proposed by Hinrichs and Nakazawa (1996) is to let lexical rules apply only to inputs which are subsumed by the input conditions of the rule. Since inherited complements are not instantiated (lexically) on COMPS of a complement inheritance verb, the complement extraction lexical can no longer extract inherited complements. This solution is not without problems, however. First, more recent versions of the CELR, such as the one proposed in Sag (1995), both instantiate and delete an element in the input. Thus, for such a rule it is crucial that the input is not necessarily subsumed by the input conditions of the rule. Second, subsumption appears to be thoroughly incompatible with processing strategies involving delayed evaluation (van Noord and Bouma 1994), a technique which is relevant especially for the type of grammar considered in Hinrichs and Nakazawa (1996). For a subsumption test, the moment of evaluation, and thus the order in which constraints are evaluated, is essential. For delayed evaluation, however, it must be the case that order in which constraints are evaluated can be determined dynamically.

The constraint-based analysis of complement extraction developed in section 2 integrates the account of extraction with the mapping between argument structure and valence. Remember that `map-non-subj-args` determines for each (non-subject) element on ARG-ST whether it is to be realized as a complement or as a *gap*. Consequently, only *arguments* of a verb can be extracted. Since the extracted NP in examples such as (11) above appears on the ARG-ST of *kaufen* only, no spurious ambiguity will arise. Thus, the elimination of lexical rules also eliminates the problem observed in Hinrichs and Nakazawa (1996), without requiring a subsumption test.

The introduction of complement inheritance does present another kind of challenge, however. In the constraint-based fragment presented above, verbal subcategorization types only specify argument structure. The mapping between argument structure and valence is determined by a general `map-args` constraint. Complement inheritance verbs are characterized by the fact that their COMPS-list may contain (inherited) complements which do not correspond to elements of ARG-ST. Consequently, complement inheritance verbs do not obey the `map-args` constraint as defined in the previous section.

Complement inheritance can be accounted for if a rather different characterization of complement inheritance is introduced. Together with a modification of the `map-args` constraint this will make it possible to include complement inheritance verbs in the constraint based fragment developed so far.

Whereas complements are normally saturated phrases (i.e. their COMPS-value is the empty list), the verbal complements of complement inheritance verbs need not be saturated. Thus, in terms of the verbal subcategorization relation introduced in the previous section, the distinction between a regular VP-complement taking verb, such as *versuchen* (*try*) and *können* is that the former requires a saturated VP whereas the latter selects for a (lexical) verbal complement, but does not impose any conditions on the value of COMPS of that complement. The relevant entries for `verbal-subcat` are given below.

$$(12)\quad \texttt{verbal-subcat}(\begin{bmatrix} \text{PHON} & \textit{versuchen} \\ \text{ARG-ST} & \langle\ \text{NP}_i,\ \text{V}_j[\text{COMPS}\ \langle\ \rangle] \rangle \\ \text{CONT} & \textit{versuchen}'(i,j) \end{bmatrix})$$

$$\texttt{verbal-subcat}(\begin{bmatrix} \text{PHON} & \textit{können} \\ \text{ARG-ST} & \langle\ \text{NP}_i,\ \text{V}_j[\text{COMPS}\ \boxed{1}\ ] \rangle \\ \text{CONT} & \textit{können}'(i,j) \end{bmatrix})$$

Note that $\boxed{1}$ in the second clause is provided only to make the contrast with the first clause explicit. As it is an anonymous variable, no constraint whatsoever is imposed on the value of COMPS. This suffices as a characterization of complement inheritance, if `map-non-subj-args` is modified as follows:

(13) $\texttt{map-non-subj-args}(\langle\ \rangle)$

$\texttt{map-non-subj-args}(\langle\ \boxed{1}\,[\textsc{comps}\ \boxed{2}\,]\ \mid\ \boxed{3}\ \rangle,\ (\boxed{2} \oplus \langle\ \boxed{1}\ \rangle \oplus \boxed{4})\ ) \leftarrow$

$\texttt{map-non-subj-args}(\boxed{3},\ \boxed{4})$

$$\texttt{map-non-subj-args}(\left\langle \begin{bmatrix} \textsc{loc} & \boxed{1} \\ \textsc{slash} & \{\ \boxed{1}\ \} \end{bmatrix} \mid\ \boxed{2} \right\rangle,\ \boxed{3}) \leftarrow$$

$\texttt{map-non-subj-args}(\boxed{2},\ \boxed{3})$

The second clause, which maps non-subject arguments onto COMPS also prepends the complements of this element. This clause applies generally (i.e. to all complements) and thus the possibility of complement inheritance is the rule, rather than the exception. Note, however, that for verbs selecting saturated complements, $\boxed{2}$ in the definition above will be the empty list. In those cases, the definition of `map-non-subj-args` simply works as before. In cases where the value of COMPS of a complement is left unspecified (i.e. the verbal complement of an inheritance verb) the definition has the effect of prepending the complements of the verbal complement on COMPS, and thus a lexical entry will result which is identical to what is proposed in Hinrichs and Nakazawa (1994).

4 Conclusions

We have argued that recursive constraints can be used to eliminate a highly problematic class of lexical rules, i.e. those affecting valence. Apart from avoiding a number of technical difficulties associated with the use of lexical rules, the constraint-based alternative has the advantage of providing a uniform account of complement and adjunct selection without spurious ambiguity.

References

Borsley, R. D. (1989). An HPSG approach to Welsh. *Journal of Linguistics 25*, 333–354.

Bouma, G. (1992). Feature structures and nonmonotonicity. *Computational Linguistics 18*(2), 183–204.

Carpenter, B. (1992). Skeptical and creduluous default unification with applications to templates and inheritance. In T. Briscoe, A. Copestake, and V. de Paiva (Eds.), *Default Inheritance within Unification-Based Approaches to the Lexicon.* Cambridge: Cambridge University Press.

Frank, A. (1994). Verb second by lexical rule or by underspecification. Technical report, Institute for Computational Linguistics, Stuttgart.

Hinrichs, E. and T. Nakazawa (1994). Linearizing AUXs in German verbal complexes. In J. Nerbonne, K. Netter, and C. Pollard (Eds.), *German in Head-driven Phrase Structure Grammar*, Lecture Note Series, pp. 11–38. Stanford: CSLI.

Hinrichs, E. and T. Nakazawa (1996). Applying lexical rules under subsumption. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, Copenhagen, pp. 543–549.

Iida, M., C. Manning, P. O'Neill, and I. Sag (1994). The lexical integrity of Japanese causatives. Paper presented at the LSA 1994 Annual Meeting.

Kathol, A. (1994). Passive without lexical rules. In J. Nerbonne, K. Netter, and C. Pollard (Eds.), *German in Head-driven Phrase Structure Grammar*, Stanford, pp. 237–272. CSLI.

Lascarides, A., T. Briscoe, N. Asher, and A. Copestake (1996). Order independent and persistent typed default unification. *Linguistics and Philosophy 19*(1), 1–89.

Manning, C. and I. Sag (1995). Dissociations between argument structure and grammatical relations. Draft, Stanford University, July 1995.

Manning, C., I. Sag, and M. Iida (1996). The lexical integrity of Japanese causatives. In T. Gunji (Ed.), *Studies on the Universality of Constraint-based Phrase Structure Grammars*. Osaka University.

Meurers, W. D. (1995). Towards a semantics of lexical rules as used in HPSG. In *Proceedings of the Conference on Formal Grammar*, Barcelona.

Meurers, W. D. and G. Minnen (1995). The covariation approach as computational treatment of HPSG lexical rules. In *Proceedings of the Fifth International Workshop on Natural Language Understanding and Logic Programming*, Lisbon.

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

Pollard, C. and I. Sag (1987). *Information Based Syntax and Semantics, Volume 1*. Center for the Study of Language and Information Stanford.

Pollard, C. and I. Sag (1994). *Head-driven Phrase Structure Grammar*. Center for the Study of Language and Information Stanford.

Sag, I. (1995). Constraint-based Extraction (Without a Trace). Draft, Stanford University, November, 1995.

Sag, I. and P. Miller (to appear). French clitic movement without clitics or movement. *Natural Language and Linguistic Theory*

van Noord, G. and G. Bouma (1994). Adjuncts and the processing of lexical rules. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, Kyoto, pp. 250–256.