Implementation of Optional Specifiers in a Lexicalist Framework

Margriet Verlinden

Tilburg University, Computational Linguistics and Artificial Intelligence Group

Abstract

When implementing a lexicalist theory like HPSG, the redundancy of lexical information can cause inefficiency during parsing. One of the main sources of this redundancy is the great number of possible subcategorization patterns that a constituent often has. A common case of this is that a constituent subcategorizes for certain elements optionally. Since optionality is not expressed in feature structures, this results in multiplication of structures, and thus can cause inefficiency in the parsing process. A way to work with single, unexpanded feature structures in such cases was therefore developed and implemented. In this approach, important use is made of the type system of a typed feature logic. The other mechanisms of HPSG, such as the features, the grammar rules and the principles remain the same. In this paper, the inefficiency problem caused by optional specifiers is described, and the solution we developed for such cases, like plural nouns and mass terms. This solution has been implemented only for the selector feature SPR, but the approach can be applied to all valence features that contain zero or one element (SPR, SUBJ, and sometimes COMP). The paper concludes with some ideas on how the method should be extended for longer valence lists as well (e.g. the COMP-list of ditransitive verbs), and how to allow optionality in general to be handled efficiently.

1 The inefficiency of empty determiners

In the DENK project (Bunt et al., 1998), a cooperative multimodal interface is developed.[1] The user can communicate in English with a so-called cooperative assistant, which helps the user to perform a certain task. For the interpretation of the user's utterances various software components have been built, including, among others, a parser, a reasoning component and a context database. The work reported here was performed when writing a grammar for the DENK parser. One of the phenomena for which an analysis was needed were the 'bare plurals', in fact, optional specifiers in general. An example of a bare plural is the word 'buttons' in sentence (1).

(1) Dangerous buttons of the microscope are red.

In HPSG-III (Pollard and Sag, 1994: ch. 9), the feature SPR indicates for nouns their need for a determiner, and the feature SPEC indicates for determiners the need for an N-bar. For NPs without a determiner, such as bare plurals and

[1] 'DENK' stands for 'Dialogue management & Knowledge acquisition'.

mass NPs, it is suggested in Pollard and Sag (1994: p. 90, fn. 25) to use phonetically empty determiners. The empty determiner ε-det has the specification [SPEC [SPR ⟨*synsem*⟩]]. For several reasons, Pollard and Sag are not happy with empty elements in HPSG, however. Moreover, phonetically empty elements are in general a major source of inefficiency in parsing, in particular in chart parsing. The problem with phonetically empty elements is that they can be assumed to be present between any two words of the input. During parsing this can result in a large number of extra items on the chart, of which generally only a small subset are used for a final parse. The parsing of sentence (1) for example, will result in the chart shown in Figure 1.

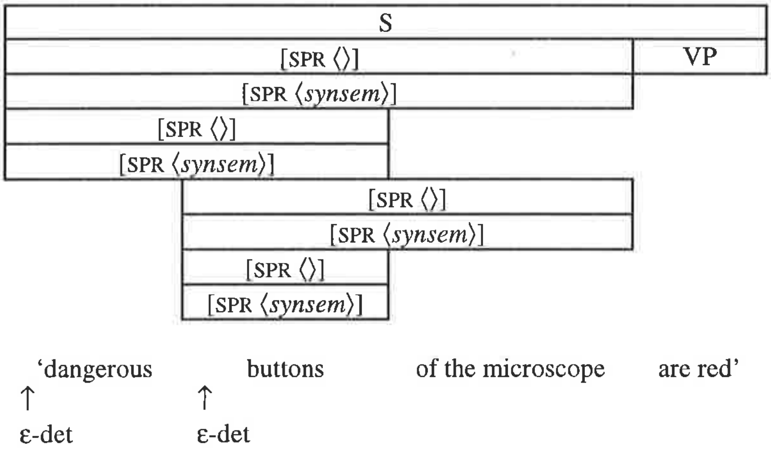

Figure 1: Unwanted duplication of items on the chart due to empty determiners.

During parsing, every possible combination of items is checked for the possibility to create a new item from it. As can be seen, two items are created for each combination of the plural noun 'buttons' and the surrounding modifiers. This is correct, because these phrases are indeed both NPs, which means they are satisfied for SPR, and at the same time they are N-bars, which implies that they are unsatisfied for SPR. If this could be represented in another way, it might reduce the number of items on the chart significantly. Because chart parsing means testing each set of items and creating a new item whenever possible, any reduction of the number of items increases the efficiency of the chart parsing process directly.

Alternative analyses in which non-branching grammar rules such as NP → N-bar(*plur*) are employed give rise to the same duplication of items.

2 An analysis without empty determiners

In order to avoid double items, we have chosen to underspecify the SPR value of plural noun phrases. In other words, we developed a way to express the optionality of a specifier. Note that this is not looking for a mere notation that can be used to indicate optionality (such as special parentheses or some token to mark the optional element) but for a technique with well-established properties within HPSG, which can be used in an implementation. We chose to use typed feature logic (Carpenter, 1992) since it offers great expressivity. As usual with typed feature logic, the main choice is to do something special either with features or with types.

2.1 Types

One possibility to express the optionality of a specifier is an extra *feature* in the element on the SPR list, indicating whether the element is obligatory or optional. Another possibility is to put this information in the *type* of the elements of the list, which is normally *synsem*. Only with the second option can a maximal efficiency gain be realized, as we will see at the end of this section. Therefore, we introduced two new subtypes of *synsem*: *opt_synsem* and *oblig_synsem*, resulting in the partial type hierarchy shown in Figure 2. The appropriate features of *synsem* remain unaltered and the new subtypes have no other appropriate features than the ones they inherit from *synsem*.

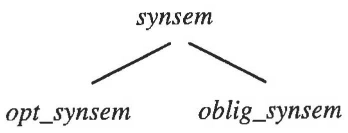

Figure 2: Extended type hierarchy of synsems.

The new *synsem*-types are to be used in a minimal number of places in the grammar and lexicon. All feature structures of parsed constituents will have SYNSEM features of type *synsem*, as before. The SYNSEM feature of the elements on any SPR list can be either of type *oblig_synsem* or of type *opt_synsem*. In the first case it expresses an obligatory specifier, in the second case an optional specifier. An N-bar with the specification [SPR ⟨*opt_synsem*⟩] can be combined with a determiner as usual, but may be regarded as satisfied for SPR as well.[2] Parsing example sentence (1) will now result in a chart which contains just one item for each combination of ‘buttons’ and the modifiers, as shown in Figure 3.

[2] After unification, the SYNSEM of the specifier phrase is of type *opt_synsem*. This may seem a bit odd, but it does not harm the analysis because the dimension of optionality is independent of the particular value that synsems have or can have.

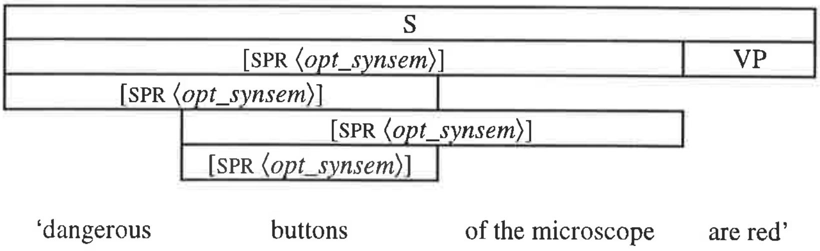

Figure 3: Chart with optional specifiers and without duplication of items.

2.2 Updating the rest of the grammar

This change in the lexical entries for nouns that subcategorize optionally for a specifier realizes the desired reduction in the number of items created during parsing. A limited number of other components of the grammar have to be altered to agree with these changes.

The grammar rules that are involved in this new analysis of noun phrases can remain the same as when the single type *synsem* is used. Neither does the relevant principle, the Valence Principle, have to be altered. Some of the SPR specifications in the lexicon need to be adapted for this analysis. The adaptations are different for various kinds of SPR specifications. We distinguish two types of those specifications: the specifications for SPR of a word on the one hand, and for MOD|SPR, SPEC|SPR, and SUBJ⟨SPR⟩, COMP⟨..,SPR,..⟩ etc. of that word on the other hand. We call these specifications henceforth respectively the *main* SPR specifications of a word and the SPR specifications *inside* the selector features of a word (MOD, SPEC and the valence features). The main specifications of words that are saturated for SPR remain the same: [SPR ⟨⟩]. Words with this specification are for example pronouns. Main specifications of the form [SPR ⟨*synsem*⟩] now have to be either [SPR ⟨*oblig_synsem*⟩], or, for any word that can occur without a determiner: [SPR ⟨*opt_synsem*⟩]. An example of the first case are singular nouns, and an example of the second case are plural nouns.

In contrast with the main specifications, [SPR ⟨*synsem*⟩] in a selecting feature can remain the same, e.g. in the MOD of an adjective. The specification [SPR ⟨⟩] in selecting features, meaning 'satisfied for SPR', now has to be [SPR ⟨⟩ ∨ ⟨*opt_synsem*⟩], e.g. in the COMP of a preposition (for PPs like 'with the red button' and 'with red buttons', respectively).

2.3 Efficient implementation

All of the above specifications for SPR can be realized in most systems for implementing typed feature structures. In the DENK grammar, these were all implemented as described, except for the one with the disjunction ([SPR ⟨⟩ ∨ ⟨*opt_synsem*⟩]). The disjunction in this last specification can be implemented, but it is undesirable to do so, because disjunctions lead to inefficiency in most implementations, as it does in the parser engine that we used (ALE; Carpenter and Penn, 1994). In order to make the proposed analysis as efficient as possible, we worked out a way to express 'satisfied for SPR' without disjunction. This was done by making the optionality of the list elements also visible from the outside of the list: in the type of the list. Therefore, the hierarchy of list types was extended. The new partial type hierarchy with most general supertype *list* is shown in Figure 4.

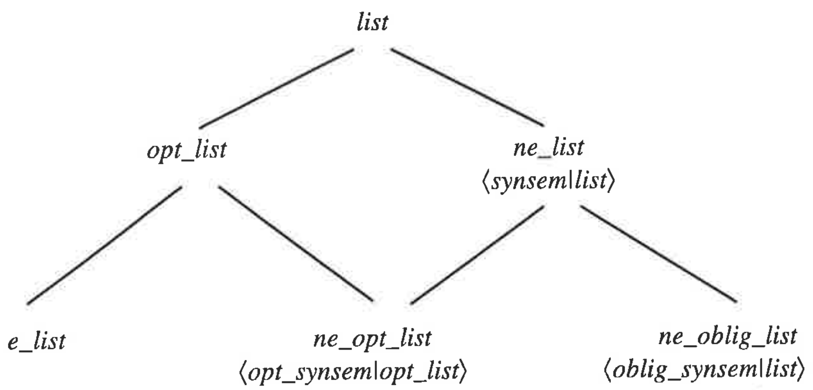

Figure 4: Extended type hierarchy of lists.

In addition to the simple distinction of *e_list* and *ne_list*, we introduced *opt_list*, a new type of list which consists exclusively of zero or more optional elements. This is exactly what we need for the disjunction '⟨⟩ ∨ ⟨*opt_synsem*⟩'. Lists of type *ne_opt_list* contain only elements of the type *opt_synsem*. Other non-empty lists (*ne_oblig_list*), now consist of one or more *oblig_synsems*, followed by zero or more *opt_synsems* (there is no interleaving of *oblig_synsems* and *opt_synsems*). In the implementation of the grammar for DENK, these properties are forced by defining the correct constraints on the appropriate features, HD and TL. The constraints are:

$$\textit{ne\_list:} \quad \begin{bmatrix} \text{HD} & \textit{synsem} \\ \text{TL} & \textit{list} \end{bmatrix}$$

$$\textit{ne\_opt\_list:} \quad \begin{bmatrix} \text{HD} & \textit{opt\_synsem} \\ \text{TL} & \textit{opt\_list} \end{bmatrix}$$

$$\textit{ne\_oblig\_list:} \quad \begin{bmatrix} \text{HD} & \textit{oblig\_synsem} \end{bmatrix}$$

The (most general) type of the feature SPR remains simply *list*.

The constraint 'satisfied for SPR' (needed in the COMP feature of prepositions and many other valence features) can now be stated without disjunction; for instance, instead of '$\langle\rangle \vee \langle opt\_synsem\rangle$' we can now write *opt_list*.[3] This optimization would not be possible if the optionality had been indicated by using a (new) feature inside the elements on the SPR-list. Only the use of types has let us create types of lists that can quickly be checked for the presence of any obligatory items, because that characteristic is part of their definition.[4]

3 The semantics

In standard HPSG, quantifiers are interpreted during parsing by means of the QSTORE mechanism. This requires all quantifiers to be present. The quantifier associated with 'green books' in sentence (2) is different from the one in sentence (3).

(2) The man sees the green books.
(3) The man sees green books.

The first would be *the*, the second *some* or *exists*. Semantically speaking, one strategy is always to derive quantifiers from determiners, corresponding to the syntactic strategy of using empty determiners. Another is to use a non-branching rule to lift N-bars to NPs, and to introduce the right quantifier. Any analysis that does not use one of these devices requires a different semantic strategy.

One possibility is to view NPs as having a two-part meaning, a quantificational part and a predicate part, and to have this reflected in their semantic representations. This strategy has been implemented for determinerless mass NPs in the TENDUM system and described by Bunt (1995). In the lexicon, nouns can be given such a two-part meaning as well, where the quantificational part corresponds to *some* and is used as a default quantifier for determinerless NPs. Another strategy, which is followed in the DENK project, is to postpone the interpretation of quantifiers altogether, both with respect to scope and collective/distributive readings, and build an underspecified logical form (ULF; Kievit, 1996). This ULF is interpreted after parsing, using information from the dialogue context. This solution seems particularly elegant because of the parallel use of underspecification in the semantics and in the syntax. Examples of ULFs that are produced by the parser of the DENK system are given below. A ULF representa-

[3] With these new types, there is also an alternative way to express the specifications that did not have disjunctions: ${}_{ne\_opt\_list}\langle\_\rangle$ for $\langle opt\_synsem\rangle$, ${}_{ne\_oblig\_list}\langle\_\rangle$ for $\langle oblig\_synsem\rangle$, etc., where $\langle\_\rangle$ stands for [TL: *e_list*]. This rewriting has no effect, however, on the efficiency of the parser. Therefore, in DENK the new list types are only used to avoid disjunctions.

[4] In this paper, the valence features are lists of synsems. In some implementations, as in the one for DENK, lists of signs are used instead. This can be the case either because the grammar is based upon HPSG-I (Pollard and Sag, 1987), or (as in DENK) for simpler, more convenient coreferencing between the valence list elements and the daughters of an ID schema. In such grammars, similar subtypes can be created for *sign* and *sign_list* as for *synsem* and *synsem_list*, and exactly the same benefits hold for the entire analysis and implementation described in this paper. Because the list of signs do not automatically imply the Locality Principle as lists of synsems do, implementations with lists of signs must guarantee this principle in another way, which is in general not a problem.

tion has four elements: a quasi-determiner, a parameter, a feature list, and a restriction list. The name 'quasi-determiner' stands for everything that determines the kind of ULF. Quasi-determiners correspond to syntactic determiners for object-ULFs. Event-ULFs are recognizable by the special quasi-determiner 'event'. For more details about ULF we refer the reader to Kievit (1996). The ULF of sentence (2) is given in (4).

(4) ulf(event,X,[tense:pres,...],
    [restr(see,X,[agent: ulf(*the*,Y,[prs:3,nm:sg],[man(Y)]),
        theme: ulf(*the*,Z,[prs:3,nm:pl],[book(Z),
            green(Z)])])])

In the ULF of a bare plural no specific quantifier is filled in. Instead, a term *quant*, which has very little meaning, is used. So the complete ULF of sentence (3) is as in (5).

(5) ulf(event,X,[tense:pres,..],
    [restr(see,X,[agent: ulf(*the*,Y,[prs:3,nm:sg],[man(Y)]),
        theme: ulf(*quant*,Z,[prs:3,nm:pl],[book(Z),
            green(Z)])])])

During parsing, *quant* means as much as: 'no quantifier has been found so far'. Any *quant* that is still present in the ULF of a complete utterance after parsing stands for 'a quantifier that belongs to an object that had no overt quantifier in its morphological representation'. This ULF is further interpreted by the reasoning component of the DENK system. The underspecified quantifier *quant* will often be interpreted as *some*, but other interpretations are also possible. In the phrase 'in high magnification mode' for example, which can typically occur in the DENK application, *quant* has to be interpreted as a definite quantifier.[5] More detail about the interpretation of ULF can be found in Kievit (1997).[6]

[5] Although not implemented in the DENK system, the choice between an existential and a generic quantifier would be made at this point, where context information is accessible.

[6] One might wonder how the same entry for 'books' can account for the correct content of both 'green books' (which are books) and 'fake books' (which are not books). With the described semantic treatment, it is however possible to define the right behaviour of non-intersective adjectives such as 'fake'. Restrictions in ULF can be simple, as 'book(Z)' in (5) and (6); they can also be complex, in the case of non-intersective adjectives. With the modifier being the semantic head, as is usual in HPSG, a complex restriction can easily be generated (second order quantification). This is illustrated with the parse tree for the NP 'fake books' in (i).

(i) 'fake books'
ulf(*quant*,Z,[prs:3,nm:pl],[fake(book,Z)])
/ \
'fake' 'books'
ulf(*quant*,Z,[],[fake(Y,X)]) ulf(*quant*,X,[prs:3,nm:pl],[book(X)])

4 Conclusion and further extensions

We have described an alternative analysis for bare plurals which is more efficient than the standard analyses when it comes to chart parsing. In our analysis we applied an elegant and useful combination of syntactic and semantic underspecification. The analysis described is not just applicable to bare plurals, but also to mass nouns or any other noun that can function as an NP either with or without a determiner.

The analysis with the selection of either optional or obligatory elements has been implemented for plural nouns and mass terms in the DENK system. Parsing sentence (6) for example, 29 instead of 35 items were created on the chart (–17%). The gain is of course bigger for sentences with more mass terms and plural nouns. Sentence (7) for example, results in 39 instead of 49 items (–20%).

(6) The red buttons of the microscope are dangerous!
(7) You can compare electron beams with light rays.

Furthermore, the approach outlined here can be used not only for optional determiners but also for other subcategorization lists of length one. Many words have, for example, an optional complement. The so-called 'picture'-nouns are considered to have a PP-complement with the preposition 'of'. However, words like 'picture' can just as well be used without this PP. In the classical HPSG treatment, there have to be two lexical entries for such words, one with an element on the COMP-list and one with an empty COMP-list. With the tools proposed in this paper, one entry would suffice, with the constraint COMP ⟨ $_{opt_synsem}$PP('of')⟩.

Finally, it might be possible to extend our approach to describe optionality of any subcategorization feature, of any size. This would be very useful, since optionality in subcategorization has up to now been an important source of inefficiency in lexical theories like HPSG. There are two remarks to be made with respect to this more general application of the analysis.

First, it must be noticed that the elements on any non-empty list in the type system given in this paper are always ordered in such a way that all the obligatory elements precede all optional elements (the order is not important for the features SUBJ and SPR because they never contain more than one element). In HPSG, the order of the elements on COMP corresponds to the obliqueness hierarchy. In the version of HPSG from Pollard and Sag (1994), which forms the basis for the DENK parser, the left-to-right order corresponds to increasing obliqueness. The order of the complements is thus: primary object, secondary object, then oblique PP and verbal and/or predicative complements. This means that the proposed adaptations can be made in all those cases where all optional complements are more oblique than the obligatory ones. This is for example the case for 'donate' as in 'donate something to somebody'. The constraint could be as in (8).

(8) 'donate': COMP $\langle$ $_{oblig_synsem}$NP, $_{opt_synsem}$PP('to')$\rangle$

However, the relation between obliqueness and optionality of complements is not entirely clear; there is some congruence between these aspects of complements, but not a total one as can be seen from examples (9) and (10).[7]

(9) Kim seems (to me) to be happy.
(10) Kim baked (me) a cake.

Second, it should be noted that, if the proposed treatment of optionality is used for subcategorization features with more than one element, the Valence Feature Principle would have to be revised. In Pollard and Sag (1994) this principle is stated as:

> "In a headed phrase, for each valence feature F, the F value of the head daughter is the concatenation of the phrase's F value with the list of SYNSEM values of the F-dtrs."

This has to be adapted so that the concatenation of two lists with a valid order results in a new list with a valid order, that is, with all obligatory elements preceding all optional elements.

References

Bunt, H., R. Ahn, R.J. Beun, T. Borghuis and K. van Overveld (1998). Multimodal Cooperation with the DENK System, in H. Bunt, R.J. Beun and T. Borghuis (eds.) *Multimodal Human-Computer Communication*, Springer Verlag, Berlin, 39-67.

Bunt, H. (1985). *Mass terms and model-theoretic semantics*, Cambridge University Press.

Carpenter, B. (1992). *The Logic of Typed Feature Structures*, Cambridge Tracts in Theoretical Computer Science 32, Cambridge University Press, New York.

Carpenter, B. and G. Penn (1994). *ALE: The Attribute Logic Engine, user's guide*, Computational Linguistics Program, Philosophy Department, Carnegie Mellon University Pittsburgh.

Kievit, L. (1996). *ULF (D(8b)-9606-spec)*, DENK Report 96/27, Samenwerkings-Orgaan Brabantse Universiteiten, Tilburg and Eindhoven.

Kievit, L. (1997). *CLUES I*, DENK Report 97/30, SamenwerkingsOrgaan Brabantse Universiteiten, Tilburg and Eindhoven.

[7] In the implementation for DENK, the order on COMP is the surface order. This corresponds in English roughly to increasing obliqueness. When the system of Linear Precedence Rules in HPSG is further elaborated — in theory and in practice — in the future, the obliqueness order can easily be (re-)established.

Pollard, C. and I. Sag (1987). *Information-Based Syntax and Semantics*, Vol. 1 Fundamentals, CSLI.

Pollard, C. and I. Sag (1994). *Head-Driven Phrase Structure Grammar*, University of Chicago Press.