

Selectional Restrictions in HPSG

Ion Androutsopoulos

Software and Knowledge Engineering Laboratory Language Technology Group
Institute of Informatics and Telecommunications Department of Computing
National Centre for Scientific Research “Demokritos”
Macquarie University
Sydney NSW 2109, Australia
153 10 Ag. Paraskevi, Athens, Greece
e-mail: ionandr@iit.demokritos.gr

Robert Dale

Department of Computing
Macquarie University
Sydney NSW 2109, Australia
e-mail: Robert.Dale@mq.edu.au

Abstract

Selectional restrictions are semantic sortal constraints imposed on the participants of linguistic constructions to capture contextually-dependent constraints on interpretation. Despite their limitations, selectional restrictions have proven very useful in natural language applications, where they have been used frequently in word sense disambiguation, syntactic disambiguation, and anaphora resolution. Given their practical value, we explore two methods to incorporate selectional restrictions in the HPSG theory, assuming that the reader is familiar with HPSG. The first method employs HPSG’s BACKGROUND feature and a constraint-satisfaction component pipe-lined after the parser. The second method uses subsorts of referential indices, and blocks readings that violate selectional restrictions during parsing. While theoretically less satisfactory, we have found the second method particularly useful in the development of practical systems.

1 Introduction

The term *selectional restrictions* refers to semantic sortal constraints imposed on the participants of linguistic constructions. Selectional restrictions are invoked, for example, to account for the oddity of (1) and (3) (cf. (2) and (4)).

- (1) ?Tom ate a keyboard.
- (2) Tom ate a banana.
- (3) ?Tom repaired the technician.
- (4) Tom repaired the keyboard.

To account for (1) and (2), one would typically introduce a constraint requiring the object of “to eat” to denote an edible entity. The oddity of (1) can then be attributed to a violation

of this constraint, since keyboards are typically not edible. Similarly, in (3) and (4) one could postulate that “to repair” can only be used with objects denoting artifacts. This constraint is violated by (3), because technicians are typically persons, and persons are not artifacts.

We note that selectional restrictions attempt to capture contextually-dependent constraints on interpretation. There is nothing inherently wrong with (1), and one can think of special contexts (e.g. where Tom is a circus performer whose act includes gnawing on computer peripherals) where (1) is felicitous. The oddity of (1) is due to the fact that in most contexts people do not eat keyboards. Similarly, (3) is felicitous in a science-fiction context where the technician is a robot, but not in most usual contexts. Selectional restrictions are typically used to capture facts about the world which are generally, but not necessarily, true.

In various forms, selectional restrictions have been used for many years, and their limitations are well-known (Allen, 1995). For example, they cannot account for metaphoric uses of language (e.g. (5)), and they run into problems in negated sentences (e.g. unlike (1), there is nothing odd about (6)).

- (5) My car drinks gasoline.
- (6) Tom cannot eat a keyboard.

Despite their limitations, selectional restrictions have proven very useful in practical applications, and they have been employed in several large-scale natural language understanding systems (Martin et al., 1986) (Alshawi, 1992). Apart from blocking pragmatically ill-formed sentences like (1) and (3), selectional restrictions can also be used in word sense disambiguation, syntactic disambiguation, and anaphora

resolution. In (7), for example, the “*printer*” refers to a computer peripheral, while in (8) it refers to a person. The correct sense of “*printer*” can be chosen in each case by requiring the object of “*to repair*” to denote an artifact, and the subject of “*to call*” (when referring to a phone call) to denote a person.

(7) Tom repaired the printer.

(8) The printer called this morning.

Similarly, (9) is from a syntactic point of view potentially ambiguous: the relative clause may refer either to the departments or the employees. The correct reading can be chosen by specifying that the subject of “*retire*” (the relativised nominal in this case) must denote a person.

(9) List the employees of the overseas departments that will retire next year.

Given the value of selectional restrictions in practical applications, we explore how they can be utilised in the HPSG theory (Pollard and Sag, 1994), assuming that the reader is familiar with HPSG. Our proposals are based on experience obtained from using HPSG in a natural language database interface (Androutsopoulos et al., 1998) and a dialogue system for a mobile robot. To the best of our knowledge, selectional restrictions have not been explored so far in the context of HPSG.

We note that, although they often exploit similar techniques (e.g. semantic sort hierarchies), selectional restrictions constitute a different topic from *linking theories* (Davis, 1996). Roughly speaking, linking theories explore the relation between thematic roles (e.g. agent, patient) and grammatical functions (e.g. subject, complement), while selectional restrictions attempt to account for the types of world entities that can fill the thematic roles.

We discuss in sections 2 and 3 two ways that we have considered to incorporate selectional restrictions into HPSG. Section 4 concludes by comparing briefly the two approaches.

2 Background restrictions

The first way to accommodate selectional restrictions in HPSG uses the CONTEXT|BACKGROUND (abbreviated here as CX|BG) feature, which Pollard and Sag (Pollard and Sag, 1994) reserve for “felicity

conditions on the utterance context”, “presuppositions or conventional implicatures”, and “appropriateness conditions” (*op cit* pp. 27, 332). To express selectional restrictions, we add qfpsoas (quantifier-free parameterised states of affairs) with a single semantic role (slot) in CX|BG.¹ For example, apart from the *eat* qfpsoa in its NUCLEUS (NUC), the lexical sign for “*ate*” (shown in (10)) would introduce an *edible* qfpsoa in BG, requiring $\boxed{2}$ (the entity denoted by the object of “*ate*”) to be edible.

$$(10) \left[\begin{array}{l} \text{PHON } \langle ate \rangle \\ \text{SS | LOC } \left[\begin{array}{l} \text{CAT } \left[\begin{array}{l} \text{HEAD } verb \\ \text{SUBJ } \langle NP \boxed{1} \rangle \\ \text{COMPS } \langle NP \boxed{2} \rangle \end{array} \right] \\ \text{CONT | NUC } \left[\begin{array}{l} \text{EATER } \boxed{1} \\ \text{EATEN } \boxed{2} \end{array} \right] \\ \text{CX | BG } \left\{ \begin{array}{l} \text{edible } [INST \boxed{2}] \end{array} \right\} \end{array} \right] \end{array} \right]$$

In the case of lexical signs for proper names (e.g. (11)), the treatment of Pollard and Sag inserts a *naming* (*namg*) qfpsoa in BG, which requires the BEARER (BRER) to be identifiable in the context by means of the proper name. (11) also requires the bearer to be a man.

$$(11) \left[\begin{array}{l} \text{PHON } \langle Tom \rangle \\ \text{SS | LOC } \left[\begin{array}{l} \text{CAT } [HEAD noun] \\ \text{CONT } \left[\begin{array}{l} \text{INDEX } \boxed{1} \\ \text{RESTR } \{ \} \end{array} \right] \\ \text{CX | BG } \left\{ \begin{array}{l} \text{namg } \left[\begin{array}{l} \text{BRER } \boxed{1} \\ \text{NAME } Tom \end{array} \right] \\ \text{man } [INST \boxed{1}] \end{array} \right\} \end{array} \right] \end{array} \right]$$

The HPSG principles that control the propagation of the BG feature are not fully developed. For our purposes, however, the simplistic *principle of contextual consistency* of Pollard and Sag will suffice. This principle causes the BG value of each phrase to be the union of the BG values of its daughters. Assuming that the lexical sign of “*keyboard*” is (12), (10)–(12) cause (1) to receive (13), that requires $\boxed{2}$ to denote an edible keyboard.

¹To save space, we use qfpsoas wherever Pollard and Sag use quantified psos. We also ignore tense and aspect here. Consult (Androutsopoulos et al., 1998) for the treatment of tense and aspect in our HPSG-based database interface.

$$\begin{array}{l}
(12) \left[\begin{array}{l} \text{PHON } \langle \textit{keyboard} \rangle \\ \text{SS} | \text{LOC} \left[\begin{array}{l} \text{CAT} \left[\begin{array}{l} \text{HEAD } \textit{noun} \\ \text{INDEX } [2] \\ \text{RESTR } \left\{ \textit{keybd} [INST [2]] \right\} \end{array} \right] \\ \text{CONT} \left[\begin{array}{l} \text{INDEX } [2] \\ \text{RESTR } \left\{ \textit{keybd} [INST [2]] \right\} \end{array} \right] \\ \text{CX} | \text{BG } \{ \} \end{array} \right] \end{array} \right] \\
(13) \left[\begin{array}{l} \text{PHON } \langle \textit{Tom, atc, a, keyboard} \rangle \\ \text{SS} | \text{LOC} \left[\begin{array}{l} \text{CAT} \left[\begin{array}{l} \text{HEAD } \textit{verb} \\ \text{SUBJ } \langle \rangle \\ \text{COMPS } \langle \rangle \end{array} \right] \\ \text{CONT} \left[\begin{array}{l} \text{QUANTS } \langle \textit{keybd} [INST [2]] \rangle \\ \text{NUC} \left[\begin{array}{l} \text{EATER } [1] \\ \text{EATEN } [2] \end{array} \right] \\ \text{cat} \left[\begin{array}{l} \text{BRER } [1] \\ \text{NAME } \textit{Tom} \end{array} \right] \end{array} \right] \\ \text{CX} | \text{BG } \left\{ \begin{array}{l} \textit{namg} \left[\begin{array}{l} \text{BRER } [1] \\ \text{NAME } \textit{Tom} \end{array} \right] \\ \textit{man} [INST [1]], \\ \textit{edible} [INST [2]] \end{array} \right\} \end{array} \right] \end{array} \right]
\end{array}$$

According to (13), to accept (1), one has to place it in a special context where edible keyboards exist (e.g. (1) is felicitous if it refers to a miniature chocolate keyboard). Such contexts, however, are rare, and hence (1) sounds generally odd. Alternatively, one has to relax the BG constraint that the keyboard must be edible. We assume that special contexts allow particular BG constraints to be relaxed (this is how we would account for the use of (1) in a circus context), but we do not have any formal mechanism to specify exactly when BG constraints can be relaxed.

Similar comments apply to (3). Assuming that the sign of “*repaired*” is (14), and that the sign of “*technician*” is similar to (12) except that it introduces a *technician* index, (3) receives a sign that requires the repairer to be a technician who is an artifact. Technicians, however, are generally not artifacts, which accounts for the oddity of (3).

$$(14) \left[\begin{array}{l} \text{PHON } \langle \textit{repaired} \rangle \\ \text{SS} | \text{LOC} \left[\begin{array}{l} \text{CAT} \left[\begin{array}{l} \text{HEAD } \textit{verb} \\ \text{SUBJ } \langle \text{NP} [1] \rangle \\ \text{COMPS } \langle \text{NP} [2] \rangle \end{array} \right] \\ \text{CONT} | \text{NUC} \left[\begin{array}{l} \text{REPAIRER } [1] \\ \text{REPAIRED } [2] \end{array} \right] \\ \text{CX} | \text{BG } \left\{ \begin{array}{l} \textit{repair} \\ \textit{artifact} [INST [2]] \end{array} \right\} \end{array} \right] \end{array} \right]$$

Let us now consider how a computer system could account for (1)–(4). For example, how

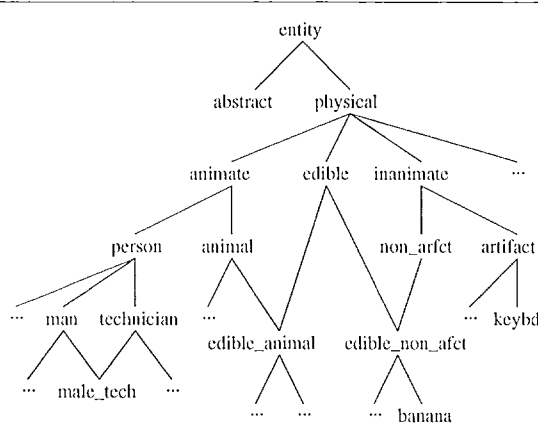


Figure 1: A simplistic semantic hierarchy

would the system figure out from (13) that (1) is pragmatically odd? Among other things, it would need to know that keyboards are not edible. Similarly, in (2) it would need to know that bananas *are* edible, and in (3)–(4) it would need to be aware that technicians are not artifacts, while keyboards are. Systems that employ selectional restrictions usually encode knowledge of this kind in the form of sort hierarchies of world entities. A simplistic example of such a hierarchy is depicted in figure 1. The hierarchy of figure 1 shows that all men and technicians are persons, all persons are animate entities, all animate entities are physical objects, and so on. Some (but not all) persons are both technicians and men at the same time; these persons are members of the *male_tech* sort. Similarly, all bananas are edible and not artifacts. No person is edible, because the sorts *person* and *edible* have no common subsorts.

It is, of course, extremely difficult to construct hierarchies that include *all* the sorts of world entities. In natural language systems that target specific and restricted domains, however, constructing such hierarchies is feasible, because the relevant entity sorts and the possible hierarchical relations between them are limited. In natural language database interfaces, for example, the relevant entity sorts and the relations between them are often identified during the design of the database, in the form of entity-relationship diagrams. We also note that large-scale semantic sort hierarchies are already in use in artificial intelligence and natural

language generation projects (for example, Cyc (Lenat, 1995) and KPML’s Upper Model (Batesman, 1997)), and that the techniques that we discuss in this paper are in principle compatible with these hierarchies.

To decide whether or not a sentence violates any selectional restrictions, we collect from the `CONT` and `BG` features of its sign ((13) in the case of (1)) all the single-role qfpsoas for which there is a sort in the hierarchy with the same name. (This rules out single-slot qfpsoas introduced by the `CONTs` of intransitive verbs.) The decision can then be seen as a constraint-satisfaction problem, with the collected qfpsoas acting as constraints. (15) shows the constraints for (1), rewritten in a form closer to predicate logic. HPSG indices (the boxed numbers) are used as variables.

$$(15) \textit{keybd}(\boxed{2}) \wedge \textit{man}(\boxed{1}) \wedge \textit{edible}(\boxed{2})$$

Given two constraints c_1, c_2 on the same variable, c_1 *subsumes* c_2 if the corresponding hierarchy sort of c_1 is an ancestor of that of c_2 or if $c_1 = c_2$. c_1 and c_2 can be replaced by a new single constraint c , if c_1 and c_2 subsume c , and there is no other constraint c' which is subsumed by c_1, c_2 and subsumes c . c and c' must be constraints on the same variable as c_1, c_2 , and must each correspond to a sort of the hierarchy. If the constraints of a sentence can be turned in this way into a form where there is only one constraint for each variable, then (and only then) the sentence violates no selectional restrictions.

In (15), $\textit{keybd}(\boxed{2})$ and $\textit{edible}(\boxed{2})$ cannot be replaced by a single constraint, because \textit{keybd} and \textit{edible} have no common subsorts. Hence, a selectional restriction is violated, which accounts for the oddity of (1). In contrast, in (2) the constraints would be as in (16).

$$(16) \textit{banana}(\boxed{2}) \wedge \textit{man}(\boxed{1}) \wedge \textit{edible}(\boxed{2})$$

$\textit{banana}(\boxed{2})$ and $\textit{edible}(\boxed{2})$ can now be replaced by $\textit{banana}(\boxed{2})$, because both subsume $\textit{banana}(\boxed{2})$, and no other constraint subsumed by both $\textit{banana}(\boxed{2})$ and $\textit{edible}(\boxed{2})$ subsumes $\textit{banana}(\boxed{2})$. This leads to (17) and the conclusion that (2) does not violate any selectional restrictions.

$$(17) \textit{banana}(\boxed{2}) \wedge \textit{man}(\boxed{1})$$

This constraint-satisfaction reasoning, however, requires a separate inferencing component that would be pipe-lined after the parser to rule out signs corresponding to sentences (or readings) that violate selectional restrictions. In the next section, we discuss an alternative approach that allows hierarchies of world entities to be represented using the existing HPSG framework, and to be exploited during parsing without an additional inferencing component.

3 Index subsorts

HPSG has already a hierarchy of feature structure sorts (Pollard and Sag, 1994). This hierarchy can be augmented to include a new part that encodes information about the types of entities that exist in the world. This can be achieved by partitioning the *ref* HPSG sort (currently, a leaf node of the hierarchy of feature structures that contains all indices that refer to world entities) into subsorts that correspond to entity types. To encode the information of figure 1, *ref* would have the subsorts *abstract* and *physical*, *physical* would have the subsorts *animate*, *edible*, *inanimate*, and so on. That is, referential indices are partitioned into sorts, and the indices of each sort can only be anchored to world entities of the corresponding type (e.g. *keybd* indices can only be anchored to keyboards).

With this arrangement, the lexical sign for “ate” becomes (18). The `BG` *edible* restriction of (10) has been replaced by the restriction that the index of the object must be of sort *edible*.

$$(18) \left[\begin{array}{l} \text{PHON } \langle \textit{ate} \rangle \\ \text{SS} | \text{LOC} \left[\begin{array}{l} \text{CAT} \left[\begin{array}{l} \text{HEAD } \textit{verb} \\ \text{SUBJ } \langle \text{NP} \boxed{1} \rangle \\ \text{COMPS } \langle \text{NP} \boxed{2} \rangle \end{array} \right] \\ \text{CONT} | \text{NUC} \left[\begin{array}{l} \text{EATER } \boxed{1} \\ \text{EATEN } \boxed{2} \textit{edible} \end{array} \right] \end{array} \right] \end{array} \right]$$

Similarly, the sign for “Tom” becomes (19) (cf. (11)), and the sign for “keyboard” introduces an index of sort *keybd* as shown in (20) (cf. (12)).

$$(19) \left[\begin{array}{l} \text{PHON } \langle \textit{Tom} \rangle \\ \text{SS} | \text{LOC} \left[\begin{array}{l} \text{CAT} \left[\begin{array}{l} \text{HEAD } \textit{noun} \\ \text{INDEX } \boxed{1} \textit{man} \\ \text{RESTR } \{ \} \end{array} \right] \\ \text{CX} | \text{BG} \left\{ \begin{array}{l} \text{BRER } \boxed{1} \\ \text{NAME } \textit{Tom} \end{array} \right\} \end{array} \right] \end{array} \right]$$

$$(20) \left[\begin{array}{l} \text{PHON } \langle \textit{keyboard} \rangle \\ \text{SS} | \text{LOC} \left[\begin{array}{l} \text{CAT} \left[\text{HEAD } \textit{noun} \right] \\ \text{CONT} \left[\begin{array}{l} \text{INDEX } \langle \textit{keybd} \rangle \\ \text{RESTR } \{ \} \end{array} \right] \\ \text{CX} | \text{BG } \{ \} \end{array} \right] \end{array} \right]$$

Unification of indices proceeds in the same manner as unification of all other typed feature structures (Carpenter, 1992). The parsing of (1) now fails, because it attempts to unify an index of sort *edible* (introduced by (18)) with an index of sort *keybd* (introduced by (20)), and no HPSG sort is subsumed by both. In contrast, the parsing of (2) would succeed, because the sign of “*banana*” would introduce an index of sort *banana*, which is a subsort of *edible* (figure 1); hence the two indices can be unified. (3) and (4) would be processed similarly.

In (7) and (8), there would be two lexical signs for “*printer*”: one introducing an index of sort *printer_person*, and one introducing an index of sort *printer_peripheral*. (*printer_person* and *printer_peripheral* would be daughters of *person* and *artifact* respectively in figure 1.) The sign for “*repaired*”, would require the index of its object to be of sort *artifact*, and the sign of “*called*” would require its subject index to be of sort *person*. This correctly admits only the reading where the repaired entity is a computer peripheral, and the caller is a person. Similar mechanisms can be used to determine the correct reading of (9).

With the approach of this section, it is also possible to specify selectional restrictions in the declarations of *qfpsoa*s in the HPSG hierarchy of feature structures, as shown in figure 2, rather than in the lexicon.² When the same *qfpsoa* is used in several lexical signs, this saves having to repeat the same selectional restrictions in each one of the lexical signs. For example, the verbs “*repair*” and “*fix*” may both introduce a *repair* *qfpsoa*. The restriction that the repaired entity must be an artifact can be specified once in the declaration of *repair* in the hierarchy of feature structures, rather than twice in the lexical signs of “*repair*” and “*fix*”.

²Additional layers can be included between *qfpsoa* and the leaf sorts, as sketched in section 8.5 of (Pollard and Sag, 1994), to group together *qfpsoa*s with common semantic roles.

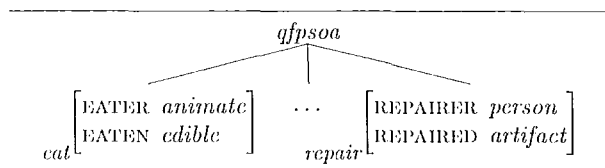


Figure 2: Declarations of *qfpsoa*s

4 Conclusions

We have presented two methods to incorporate selectional restrictions in HPSG: (i) expressing selectional restrictions as BACKGROUND constraints, and (ii) employing subsorts of referential indices. The first method has the advantage that it requires no modification of the current HPSG feature structures. It also maintains Pollard and Sag’s distinction between “literal” and “non-literal” meaning (expressed by CONT and BACKGROUND respectively), a distinction which is blurred in the second approach (e.g. nothing in (18) shows that requiring the object to denote an edible entity is part of the non-literal meaning; cf. (10)). Unlike the first method, however, the second approach requires no additional inferencing component for determining when selectional restrictions have been violated. With sentences that contain several potentially ambiguous words or phrases, the second approach is also more efficient, as it blocks signs that violate selectional restrictions during parsing. In the first approach, these signs remain undetected during parsing, and they may have a multiplicative effect, leading to a large number of parses, which then have to be checked individually by the inferencing component. We have found the second approach particularly useful in the development of practical systems.

There is a deeper question here about the proper place to maintain the kind of information encoded in selectional restrictions. The applicability of selectional restrictions is always context-dependent; and for any selectional restriction, we can almost always find a context where it does not hold. Our second method above effectively admits that we cannot develop a general purpose solution to the problem of meaning interpretation, and that we have to accept that our systems always operate in specific contexts. By committing to a particular context of interpretation, we ‘compile into’ what was traditionally thought of as literal meaning a

set of contextually-determined constraints, and thus enable these constraints to assist in the HPSG language analysis without requiring an additional reasoning component. We take the view here that this latter approach is very appropriate in the construction of real applications which are, and are likely to be for the foreseeable future, restricted to operating in limited domains.

References

- J.F. Allen. 1995. *Natural Language Understanding*. Benjamin/Cummings.
- H. Alshawi, editor. 1992. *The Core Language Engine*. MIT Press.
- I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. 1998. Time, Tense and Aspect in Natural Language Database Interfaces. *Natural Language Engineering*, 4(3):229–276.
- J.A. Bateman. 1997. Enabling Technology for Multilingual Natural Language Generation: the KPML Development Environment. *Natural Language Engineering*, 3(1):15–55.
- B. Carpenter. 1992. *The Logic of Typed Feature Structures*. Number 32 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.
- T. Davis. 1996. *Lexical Semantics and Linking in the Hierarchical Lexicon*. Ph.D. thesis, Stanford University.
- D.B. Lenat. 1995. CYC: A Large-Scale Investment in Knowledge Infrastructure. *Communications of ACM*, 38(11):33–38.
- P. Martin, D. Appelt, and F. Pereira. 1986. Transportability and Generality in a Natural-Language Interface System. In B. Grosz, K. Sparck Jones, and B. Webber, editors, *Readings in Natural Language Processing*, pages 585–593. Morgan Kaufmann.
- C. Pollard and I.A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press and Center for the Study of Language and Information, Stanford.