Instructions to Authors

MANUSCRIPTS The normal language of publication is English, but articles in French and German may also be submitted. Authors not writing in their mother-tongue are asked to have their manuscripts checked for errors in language and style.

The suggested length of articles is 20–30 pages (37,500–56,500 characters), and of book reviews 5–10 pages (9,500–19,000 characters). All manuscripts must be typewritten with double spacing, on one side of the paper only, using a conventional size of paper, preferably A4, and with generous left margins. Separate sheets should be used for: 1) Title page, with author's name(s), 2) Abstract, 3) Tables, 4) Figure legends, 5) Notes, 6) References. The *original* manuscript should be submitted, with two extra copies. *The author should retain a copy.* Forms in a language not written with the Latin alphabet must be transliterated or transcribed. Whenever possible authors should use standard phonetic transcriptions (preferably IPA) and avoid other signs. Cited forms of the language discussed should be underlined (italics); phonetic representation should be included in square brackets [. . .], phonemic representation between slant lines /. . ./, glosses should be added directly after the cited form between single quotes, and reconstructed and non-occurring (ungrammatical) forms should have an asterisk. If special signs are necessary, the editor should be consulted about the printing possibilities.

ILLUSTRATIONS All illustrations and diagrams other than Plates and Tables are to be considered as Figures. Line drawings should be drawn with black India ink, in a size allowing for reduction. Photographs should be unmounted glossy enlargements showing clear details.

ABSTRACTS For articles, but not book reviews, authors should provide an abstract in English. It should not exceed 120 words and it should be typed on a separate sheet. The author's name and initial(s), year, title of article, journal title (*Nordic Journal of Linguistics*), and volume number should precede the Abstract. Example: Hovdhaugen, E. 1977. Editorial Statement. *Nordic Journal of Linguistics* 1, 1–2. The Abstract should be followed by the author's name(s) and the postal address(es) to which correspondence should be sent.

NOTES Notes should be reduced to a minimum and whenever possible incorporated in the text, if necessary in parentheses. Phonetic or phonemic transcriptions should not occur in notes. Notes should be typed with double spacing on separate sheets and numbered consecutively.

REFERENCES References to the literature should be incorporated in the text unless they relate specifically to subject matter dealt with within a note. A strictly corresponding reference list should be included at the end of the article. *References are the author's responsibility*, and the Journal style shown below should be carefully followed to save time and correspondence.

Examples in text: As Bach (1968:37) and Binnick et al. (1969:137–198) have shown . . .

Several authors (Bach 1970:37, Bach & Harms 1968:37–39, Binnick et al. 1969:137–198) have described . . .

Reference list (alphabetical, unnumbered): *Books:* Author's last name, initials, year, period. *Title underlined* (italics) and capitalized (doctoral dissertations and unpublished manuscripts should not be in italics). Place of publication, publisher. *Journals:* Author's last name, initials, year, period. Title of article (capitalized), *title of Journal* (underlined and not abbreviated), volume number, page numbers.

Examples:

Chomsky, N. 1957. *Syntactic Structures.* The Hague: Mouton.

Jacobs, R. A. & Rossenbaum, P. S. 1970. *Readings in English Transformational Grammar.* Waltham, Mass.: Ginn.

Dahl, Ö. 1970. Some Notes on Indefinites. *Language* 46, 33–41.

Hintikka, J. 1972. The Semantics of Modal Notions and the Indeterminacy of Ontology. In Davidson, D. & Harman, G. (eds), *Semantics of Natural Languages*. Dordrecht: D. Reidel, pp. 398–414.

PROOFS The author will receive one copy of the first proof, which, duly corrected, should be returned to the publisher within the specified time. Authors are requested to check their manuscripts most carefully since corrections in proof are very expensive. *The author will be charged for corrections against his/her original text.*

OFFPRINTS Thirty offprints of each article and review article and ten offprints of shorter reviews will be supplied free of charge. Additional offprints can be ordered.

Nordic Journal of Linguistics

Contents

Vol. 24, No. 2 2001

0332-5865(200111)24:2;1-2

ISSN 0332-5865

the Interpretation of Language. Berlin, New York: Walter de Gruyter, pp. 303–323. Reprinted in Link, Godehard. 1998. *Algebraic Semantics in Language and Philosophy*: CSLI lecture notes No. 74. Stanford, CA: CSLI Publications, pp. 11–34.

Montague, R. 1973. The Proper Treatment of Quantification in Ordinary English. In Hintikka, K. J. J., Moravcsik, J. M. E. & Suppes, P. (eds), *Approaches to Natural Language*. Dordrecht: Reidel, pp. 221–242. Reprinted in Montague 1974, pp. 247–270

Parsons, T. 1985. Underlying Events in the Logical Analysis of English. In LePore, E. & McLaughlin, B. (eds), *Actions and Events: Perspectives of the Philosophy of Donald Davidson*. Oxford: Blackwell, pp. 235–267.

Parsons, T. 1990. *Events in the Semantics of English. A Study in Subatomic Semantics*. Cambridge, MA: MIT Press.

Partee, B. 1997. Genitives – a Case Study [Appendix to Theo Janssen's Chapter "Compositionality"]. In Benthem, J. v. & Meulen, A. T. (eds), *Handbook of Logic and Language*. Amsterdam: Elsevier, pp. 464–470.

Partee, B. H. & Borschev, V. B. 1998. Integrating Lexical and Formal Semantics: Genitives, Relational Nouns, and Type-Shifting. In Cooper, R. & Gramkrelidze, T. (eds), *Proceedings of the 2nd Tbilisi Symposium on Language, Logic, and Computation*. Tbilisi: Center on Language, Logic, Speech, Tbilisi State University, pp. 229–241.

Partee, B. H. & Borschev, V. B. 2000. Genitives, Relational Nouns, and the Argument-Modifier Distinction. In Fabricius-Hansen, C., Lang, E. & Maienborn, C. (eds), *ZAS Papers in Linguistics 17*. Berlin: Zentrum für Allgemeine Sprachwissenschaft, Sprachtypologie und Universalienforschung, pp. 177–201.

Partee, B. H. & Borschev, V. 2001. Some Puzzles of Predicate Possessives. In Kenesei, I. & Harnish, R. M. (eds), *Perspectives on Semantics, Pragmatics and Discourse. A Festschrift for Ferenc Kiefer*. Amsterdam: John Benjamins, pp. 91–117.

Pustejovsky, J. 1993. Type Coercion and Lexical Selection. In Pustejovsky, J. (ed.), *Semantics and the Lexicon*. Dordrecht: Kluwer, pp. 73–94.

Pustejovsky, J. 1995. *The Generative Lexicon*. Cambridge, Mass.: MIT Press.

Rothstein, S. (ed.) 1998. *Events and Grammar*, vol. 70. *Studies in Linguistics and Philosophy*. Dordrecht: Kluwer Academic Publishers.

Tenny, C. & Pustejovsky, J. (eds) 2000. *Events as Grammatical Objects: The Converging Perspectives of Lexical Semantics, Logical Semantics and Syntax*, vol. 100. *CSLI Lecture Notes*. Stanford: CSLI Publications.

Vikner, C. & Jensen, P. A. 1999. A Semantic Analysis of the English Genitive: Interaction of Lexical and Formal Semantics. Ms. Copenhagen and Kolding, Denmark.

Williams, E. 1981. Argument Structure and Morphology. *The Linguistic Review* 1, 81–114.

On Expressing Lexical Generalizations in HPSG

W. Detmar Meurers

Meurers, W. D. 2001. On expressing lexical generalizations in HPSG. *Nordic Journal of Linguistics*, 24, 161–217.

This paper investigates the status of the lexicon and the possibilities for expressing lexical generalizations in the paradigm of Head-Driven Phrase Structure Grammar (HPSG). We illustrate that the architecture readily supports the use of implicational principles to express generalizations over a class of word objects. A second kind of lexical generalizations expressing relations between classes of words is often expressed in terms of lexical rules. We show how lexical rules can be integrated into the formal setup for HPSG developed by King (1989, 1994), investigate a lexical rule specification language allowing the linguist to only specify those properties which are supposed to differ between the related classes, and define how this lexical rule specification language is interpreted. We thereby provide a formalization of lexical rules as used in HPSG.

W. Detmar Meurers, Department of Linguistics, The Ohio State University, 222 Oxley Hall, 1712 Neil Avenue, Columbus OH 43210-1298, USA. E-mail: dm@ling.osu.edu

The lexicon plays a prominent role in the paradigm of Head-Driven Phrase Structure Grammar (HPSG) (Pollard & Sag 1994), a linguistic framework which assumes information-rich lexical representations and emphasizes the role of lexical generalizations. Following Flickinger (1987) one can distinguish two kinds of regularities within the lexicon: one is sometimes referred to as *vertical*, the other as *horizontal*. Vertical generalizations express that certain properties are common to all words of a single class or subclass. For example, in Pollard and Sag (1994) all finite verbs are taken to lexically assign nominative case to their subject. Horizontal generalizations, on the other hand, express a "systematic relationship holding between two word classes, or, more precisely, between the members of one class and the members of another class" (Flickinger 1987:105). A common example for such a horizontal regularity is the relationship between active verbs and their passive counterparts (cf., e.g., Bresnan 1982, Pollard & Sag 1987).

In this paper, we discuss how these two kinds of lexical generalizations can be expressed in HPSG as formalized by the *Speciate Re-entrant Logic (SRL)* of King (1989, 1994) and show how that formal setup can be extended to include lexical rules as a means for expressing horizontal generalizations. We motivate and specify a lexical rule specification language and define how it is formally interpreted in terms of

King's formal setup, thereby providing a formalization of lexical rules for the HPSG paradigm.

1. VERTICAL GENERALIZATIONS

1.1. *Abbreviations and their theoretical irrelevance*

Vertical generalizations are often encoded by some mechanism which allows the abbreviation of a lexical specification (macros, templates, frames, etc.). Once an abbreviation is defined, it can be used in the specification of each lexical entry in a class. By defining an abbreviation in such a way that it refers to an already defined one, it is possible to organize abbreviations for lexical specification in a hierarchical fashion. According to Pollard (p.c.) this is the setup that was assumed to underlie the so-called lexical hierarchy discussed in Pollard and Sag (1987: ch. 8.1). Since the method allows for a compact specification of the lexicon, it is widely used for grammar implementation. Furthermore, the use of abbreviations in the presentation of a theory or the discussion of example analyses can serve the expository purpose of focusing the reader's attention on those aspects of the theory which are central to the discussion.

From a theoretical perspective, macros are far less useful. Starting with the formalism as such, macros are not part of the formal setup of HPSG provided in King (1989, 1994). However, in Richter (1997, 1999, 2000) and Richter et al. (1999) the setup of King's SRL is extended with relations. The resulting *Relational Speciate Re-entrant Language (RSRL)* makes it possible to refer to the argument of a relation instead of having to repeat the bundle of specifications used in defining it.[1] But even if a formalization of macros were provided, what impact can abbreviations have on the adequacy of a theory? Let us first consider the question of observational adequacy of a theory, i.e., whether a particular theory licenses the grammatical signs of a particular language and rules out the ungrammatical ones. An abbreviation and the set of descriptions which are abbreviated describe the same objects. A theory written down using abbreviations and the same theory written down without them thus make exactly the same predictions. In other words, the use of abbreviations makes no difference regarding observational adequacy. While it could be argued that observational adequacy has been neglected in the generative tradition, it remains the central empirical criterion distinguishing linguistic theories for a particular language. In fact, observational adequacy has played a central role for the work in the HPSG paradigm, which has largely focused on the explicit empirical characterization of particular languages as a necessary first step towards achieving

descriptive or explanatory adequacy. In conclusion, for most of the work in the HPSG paradigm, abbreviations play no theoretical role.

Regarding more abstract levels of adequacy, the potential role of abbreviations is less transparent. Descriptive adequacy can be understood as empirical adequacy of a parameterized core theory across languages. While Pollard and Sag (1994:14) are explicit in stating that they "take it to be the central goal of linguistic theory to characterize what it is that every linguistically mature human being knows by virtue of being a linguistic creature, namely, universal grammar", an investigation of what constitutes the universal core of an HPSG grammar and how this can be parameterized for a specific language as far as we see has largely been postponed until more elaborate observationally adequate theories of particular languages have been established – and we believe this to be a very reasonable choice. But with the mid-term goal of developing a descriptively adequate theory in mind, one could use macros in the formulation of current theories as a placeholder abstracting over language specific realizations. For example, when formulating some principle restricting finite sentences, one could use a macro *S-fin* as the antecedent of a principle to abstract away from the possible realizations of finite clauses in different languages. Taking descriptive adequacy seriously would, however, require replacing such a use of macro placeholders with proper parameters as part of a meta-theory[2] of universals and parameters in an HPSG architecture of grammar.

Macros also fail to express vertical generalizations with respect to the notion of a lexical class that was at the basis of the original idea of vertical generalizations. The problem is that when one uses macros, the criterion determining which elements belong to a specific lexical class over which some generalization is to be expressed is not part of the grammar. Whether an abbreviation is used in the specification of lexical entries and where this is done is decided by the grammar writer on the basis of personal preference or some kind of meta regime which (s)he follows in writing the grammar, but it does not follow from anything in the grammar itself.[3] That no generalization in a theoretically meaningful sense is expressed can be seen from the fact that no predictions which could potentially be proven to be incorrect are made by such an encoding. Assume that some word does not obey the restrictions encoded in the abbreviation which is intended to capture the properties of its class (and thus normally is used in the specification of lexical entries licensing the words in that class). Nothing in the grammar requires us to use the abbreviation in the lexical entry of the problematic word, i.e., no conflict arises from providing a lexical entry for the problematic word.

Finally, due to the theory-external role of abbreviations, a possibly present hierarchical structure of the abbreviations is not reflected in the theory either. The hierarchical structure of abbreviations stands in no formal relationship to the hierarchical organization of types in the linguistic ontology as defined in the type hierarchy of an HPSG grammar.

1.2. *Lexical principles*

A mechanism for expressing vertical lexical generalization needs to be able to encode implicational statements of the form: If a word is described by *D*, then it also has to be described by *E* in order to be grammatical. Crucially, this expresses a generalization over all objects described by the antecedent that can be falsified if one finds grammatical linguistic objects which satisfy the antecedent but violate the consequent. Such a mechanism is readily available in the HPSG architecture assumed in Pollard & Sag (1994), where implicational constraints are the normal method used to express generalizations about phrases, such as the Head Feature Principle.

But which kind of antecedents are to be used as antecedents of the principles encoding the vertical lexical generalizations? The antecedent of a lexical principle can be any description specifying the set of words to which the generalization is supposed to apply, for example, the conjunctive description of all words which are verbal and have a finite verb form. If it turns out that the linguistic ontology on which the theory is based is not rich enough to pick out all and only those words which a generalization is supposed to apply to, the signature[4] declaring this ontology needs to be extended.[5] The idea to introduce such missing class-distinguishing properties as ordinary types is already discussed by Riehemann (1993:56) as an alternative to the 'lexical types' of Pollard and Sag (1987: ch. 8.1) conceived as a hierarchy of abbreviations.

The already present or newly introduced properties which are referred to in the antecedent in order to single out the relevant class of elements are an explicit part of the linguistic ontology, i.e., the model. An attempt to avoid falsification of a generalization encoded in a lexical principle would therefore have an observable effect on grammar denotation since it would require changing those properties of an object which cause it to be picked out as part of the specific class a principle applies to. This contrasts with the abbreviation setup discussed above, where one can avoid falsification of a supposed generalization by not using the macro in the problematic case without changing the denotation of a grammar.

Let us illustrate the idea of lexical principles with an example. For English, Pollard and Sag (1994:30) propose to assign nominative case to the subject of finite verbs as part of their lexical entries. Instead of specifying in the lexical entry of each finite verb with a nominal subject that the subject bears nominative case, one could formulate a lexical principle to ensure nominative case assignment as a generalization over all

such verbs. To do so, we first need to check whether the ontology assumed by Pollard and Sag (1994:396 ff. and ch. 9) is rich enough to single out the set of words which are verbs that have a finite verb-form and subcategorize for a nominal subject. The type *word* is introduced as a subtype of *sign*, and the different categories of signs are represented by subtypes of *head*. The *head* subtype *verb* has the additional attribute VFORM with *finite* as one of its appropriate values. Note that these distinctions are encoded under *head* in order to make them subject to the Head Feature Principle which percolates the *head* information along the head projection. Finally, the subcategorization requirements are encoded by the VALENCE attributes (which are appropriate for *category* objects to ensure that they are mediated as part of an unbounded dependency construction). The particular valence attribute SUBJ allows us to refer to the subject, so that together with the *head* subtype *noun* we can single out verbs with nominal subjects. The independently motivated ontology defined by Pollard and Sag (1994) thus is rich enough to single out the relevant subclass of words we want to generalize over. We can therefore proceed to formulate the simple lexical principle in Fig. 1 to express the generalization that nominative case is assigned to the subject requirement of each finite verbal word which has a nominal subject.[6]

$$\begin{bmatrix} \textit{word} \\ \textsc{synsem}|\textsc{loc}|\textsc{cat} \begin{bmatrix} \textsc{head} \begin{bmatrix} \textit{verb} \\ \textsc{vform}\ \textit{finite} \end{bmatrix} \\ \textsc{val}|\textsc{subj} \left\langle \begin{bmatrix} \textsc{loc}|\textsc{cat}|\textsc{head}\ \textit{noun} \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} \textsc{synsem}|\textsc{loc}|\textsc{cat}|\textsc{val}|\textsc{subj} \left\langle \begin{bmatrix} \textsc{loc}|\textsc{cat}|\textsc{head}|\textsc{case}\ \textit{nominative} \end{bmatrix} \right\rangle \end{bmatrix}$$

Fig. 1. A lexical principle assigning nominative case.

Complex vs. type antecedents. The sketched approach of expressing lexical generalizations with lexical principles (Meurers 1997) bears a lot of similarities to the principles in the work of Sag (1997), who subclassifies phrasal types and uses principles to express generalizations about nonlocal specification. It also is very similar to the lexical generalizations expressed in Bouma et al. (2001). One formal difference between their and our approach is that they only make use of type antecedents, whereas we employ complex descriptions as antecedents of the lexical principles. This difference deserves some attention since a significant part of the more recent HPSG literature seems to be limiting itself to the use of principles with type antecedents – even though, as far as we are aware, no argument has ever been made as to why such a setup would be preferable. Quite to the contrary, as we show below, there are clear advantages to using complex antecedents.

From a formal perspective, implicational constraints with complex antecedents and those with type antecedents are both well-formed expressions of the HPSG description language defined in King (1989, 1994) and they are interpreted in the same way as any other formula of that language: as the set of objects described by that formula. In particular, using implicational statements with complex antecedents does not require something additional, like a conversion into a disjunctive normal form, in order to be interpreted.

From a linguistic perspective, we believe that complex antecedents of implicational constraints are advantageous since they make it possible to use the articulate data structure of HPSG to refer to the relevant subset of objects for which some generalization is intended to be expressed. Restricting oneself to type antecedents, one needs to introduce types for every set of objects to which a generalization applies, which duplicates specifications in case the information was already encoded under one of the feature paths for independent linguistic reasons.

Take, for example, the simple lexical principle we defined in Fig. 1 to express the generalization that nominative case is assigned to the subject of finite verbs which select a nominal subject. We saw above that each of the specifications used in the complex antecedent to single out the relevant subclass of words refers to an independently motivated part of the already defined ontology. If one instead wants to use a type antecedent for this principle, one has to introduce new subtypes of *word* that duplicate the ontological distinctions which are already encoded elsewhere in the ontology for well-motivated and still applicable reasons. More concretely, one needs to introduce a type *verbal-word* as one of the subtypes of *word* and this new type must have a type like *finite-verbal-word* as one of its subtypes. Furthermore, one has to separate those finite verbal words which have a nominal subject from those which do not, so that *finite-verbal-word* has to have *finite-verbal-word-with-nominal-subject* as one of its subtypes.[7] Additionally one has to introduce (at least) three further subtypes to represent each of the other possibilities, i.e., *non-verbal-word, non-finite-verbal-word*, and *finite-verbal-word-without-nominal-subject*. Apart from having to introduce these six types lacking independent motivation, one also has to specify a principle for each type as shown in Fig. 2 to ensure that the independently motivated and required ontological distinctions encoded elsewhere in a sign, which the new subtypes are supposed to duplicate, are actually associated with the respective new subtype.

The problem which arises at this point is that even though the principles in Fig. 2 ensure that, for example, each object of type *verbal-word* bears the relevant specification of its head type, nothing enforces that every object described by $\begin{bmatrix}\textit{word}\\ \text{SYNSEM|LOC|CAT|HEAD } \textit{verb}\end{bmatrix}$ is also described by the type *verbal-word*. To enforce this, two things are required: Firstly, one has to share the (standard SRL) assumption that the most specific subtypes partition the entire domain, which is sometimes called the closed-world assumption (Gerdemann & King 1994, Gerdemann 1995). And second, one has to define principles associating the sister types of the newly introduced types with properties which are incompatible with those associated with the newly introduced types themselves.[8] For our example, this means one additionally has to define the three principles in Fig. 3.[9]

$$\textit{verbal-word} \rightarrow \begin{bmatrix}\textit{word}\\ \text{SYNSEM|LOC|CAT|HEAD } \textit{verb}\end{bmatrix}$$

$$\textit{finite-verbal-word} \rightarrow \begin{bmatrix}\textit{verbal-word}\\ \text{SYNSEM|LOC|CAT|HEAD|VFORM } \textit{fin}\end{bmatrix}$$

$$\textit{finite-verbal-word-with-nominal-subject} \rightarrow \begin{bmatrix}\textit{finite-verbal-word}\\ \text{SYNSEM|LOC|CAT|VAL|SUBJ } \left\langle\begin{bmatrix}\text{LOC|CAT|HEAD } \textit{noun}\end{bmatrix}\right\rangle\end{bmatrix}$$

Fig. 2. Principles needed to ensure the new subtypes are properly associated with the duplicated ontological distinctions.

$$\textit{non-verbal-word} \rightarrow \begin{bmatrix}\textit{word}\\ \text{SYNSEM|LOC|CAT|HEAD } \neg\, \textit{verb}\end{bmatrix}$$

$$\textit{non-finite-verbal-word} \rightarrow \begin{bmatrix}\textit{verbal-word}\\ \text{SYNSEM|LOC|CAT|HEAD|VFORM } \neg\, \textit{fin}\end{bmatrix}$$

$$\textit{finite-verbal-word-without-nominal-subject} \rightarrow \begin{bmatrix}\textit{finite-verbal-word}\\ \text{SYNSEM|LOC|CAT|VAL|SUBJ } \neg\left\langle\begin{bmatrix}\text{LOC|CAT|HEAD } \textit{noun}\end{bmatrix}\right\rangle\end{bmatrix}$$

Fig. 3. Additional principles needed to ensure the new subtypes are properly implied by the duplicated ontological distinctions.

At this point one finally has the type *finite-verbal-word-with-nominal-subject* available to describe the same set of objects as the antecedent of the principle we saw in Fig. 1. The same principle can now be expressed with a type antecedent as shown in Fig. 4.

$$\textit{finite-verbal-word-with-nominal-subject} \rightarrow \begin{bmatrix}\text{SYNSEM|LOC|CAT|VAL|SUBJ } \left\langle\begin{bmatrix}\text{LOC|CAT|HEAD|CASE } \textit{nominative}\end{bmatrix}\right\rangle\end{bmatrix}$$

Fig. 4. The principle assigning nominative case with a type antecedent.

Concluding the discussion of the example, we believe it clearly demonstrates that a setup including principles with complex antecedents has significant advantages over one employing only type antecedents. A restriction to type antecedents entails a substantial duplication of ontological distinctions which for well-motivated reasons are encoded elsewhere in the ontology, and it makes it necessary to define special principles correlating the new types with the duplicated properties.

Surfacing from the discussion of particular encodings of vertical generalizations at this point, we showed that the HPSG architecture readily supplies the formal ingredients necessary to express vertical generalizations as implicational constraints. In the main part of the paper we therefore concentrate on the formally less developed field of horizontal generalizations.

2. HORIZONTAL GENERALIZATIONS

Lexical rules are a powerful tool for capturing horizontal generalization in the lexicon (Carpenter 1991) and they are widely used in linguistic proposals expressed in the HPSG architecture. However, while a formal foundation for basic HPSG theories is provided by King (1989, 1994), until recently no such formal basis had been given to lexical rules.[10] In this paper we want to investigate the fundamental question: What are lexical rules as they are commonly written down in HPSG supposed to mean? Based on our previous work (Meurers 1994, 1995, 2000, Meurers & Minnen 1997), this paper provides an answer to this question.[11]

A second question, which also deserves to be answered if lexical rules are to play a theoretically interesting role in linguistics concerns the powerful nature of lexical rules mentioned above: What are linguistically motivated restrictions on the range of possible lexical rules? Or more concretely: What generalizations holding across lexical rules are there and how can they be expressed? While the answers to these two questions are beyond the scope of this paper, the question of generalizations across lexical rules and methods for expressing these is closely tied to the way in which lexical rules are formalized. At the end of introducing the formal basis of our lexical rule proposal in section 3.2.2, we therefore show how this formalization of lexical rules makes it possible to express generalizations over lexical rules in a straightforward way.

Lexical rules in the HPSG literature usually look like the one shown in Fig. 5, which is modeled after the rule proposed by Pollard and Sag (1987:215) to relate passive and active verbs. Note that we use the $\mapsto$ operator for lexical rules to distinguish them from the lexical principles using implication ($\rightarrow$) as discussed in the last section.

On an intuitive level, the effect that this rule is supposed to have is clear: anything in the grammar that corresponds to the AVM on the left-hand side of the rule should get related to something that corresponds to the AVM on the right-hand side. So why is this intuitive understanding not sufficient? To begin with, the rule in Fig. 5 just consists of two

$$\left[\textsc{category}\ \left[\begin{array}{ll}\textsc{head} & \left[\begin{array}{l}\textit{verb}\\ \textsc{vform}\ \textit{psp}\end{array}\right]\\ \textsc{val} & \left[\begin{array}{ll}\textsc{subj} & \langle \mathrm{NP}_{\boxed{1}}\rangle\\ \textsc{comps} & \langle \mathrm{NP}_{\boxed{2}} \mid \boxed{3}\rangle\end{array}\right]\end{array}\right]\right] \mapsto \left[\textsc{category}\ \left[\begin{array}{ll}\textsc{head} & \left[\begin{array}{l}\textit{verb}\\ \textsc{vform}\ \textit{pas}\end{array}\right]\\ \textsc{val} & \left[\begin{array}{ll}\textsc{subj} & \langle \mathrm{NP}_{\boxed{2}}\rangle\\ \textsc{comps} & \boxed{3} \oplus \big\langle (\mathrm{PP}[\mathrm{by}]_{\boxed{1}})\big\rangle\end{array}\right]\end{array}\right]\right]$$

Fig. 5. A passive lexical rule.

AVMs separated by an arrow, but a lexical rule is supposed to be some kind of *relation*. Is there any systematic way to specify what relation the notation in Fig. 5 denotes? An answer to this question presupposes a discussion of the following subquestions: First, what does it mean to "correspond" to the left-hand side of the rule? Is the input required to be as specific as the AVM on the left-hand side or is it sufficient for the input not to contain incompatible specifications?

Second, given some input to the rule, what should the corresponding output be? Intuitively, of course, it is supposed to look something like the right-hand side of the rule. But most linguists agree that it should not look exactly like the right-hand side; it is also supposed to retain some of the properties of the input. Sometimes what is intended is explained informally in the following way: change the input only in ways that the right-hand side of the rule tells us to change it, and leave everything else the same. But the right-hand side of the rule is not an algorithm; it's only a description. How are we supposed to know what this piece of syntax is telling us to do to the inputs? And, are ordinary AVMs enough to express intended changes to the input in a compact and unambiguous way?

Third, what kinds of things are the inputs and outputs to lexical rules? That is, most linguists agree that a lexical rule is some kind of relation, but what exactly does it relate? Pollard and Sag (1994) state that lexical rules are relations between lexical entries, which are descriptions of sets of words, and this is the line pursued in Calcagno (1995). Meurers (1995), however, argues that lexical rules are better treated as relations between the objects that lexical entries denote, i.e., as relations between words. And indeed some passages in Pollard and Sag (1994) seem only to be consistent with this latter approach. So which approach captures the intentions, if any?

Finally, assuming that we arrive at satisfactory answers to all of the above, how can lexical rules be integrated into a grammar in such a way as to license the desired relationships among lexical elements. That is, if lexical rules relate lexical entries, then what is the proper place in the grammar for meta-rules of this type? And if lexical rules relate word objects, how can a lexicon including lexical rules be expressed as part of the theory?

In the following, we propose one set of answers to the above questions in the hope that the lexical rule specification language and its interpretation which we define provides a sensible formalization for lexical rules as they are commonly used in HPSG.

3. THE LEXICON IN THE HPSG ARCHITECTURE

Generally speaking, a grammar in the frameworks of GB, LFG, GPSG, and early versions of HPSG includes a way to license constituent structure and a lexicon licensing the words grounding the recursion. The lexicon often is highly specified and information-rich, so that the question naturally arises as to whether the information within the lexicon can be structured in such a way as to capture generalizations about classes of words with common behavior or to eliminate redundant specification across entries. Lexical rules have been used to express such generalizations.

In the last decade, however, as the logical foundations of HPSG have been explicated in more detail (King 1989, 1994), the notion of a grammar has been simplified to a point where, from a formal point of view, no distinction is made between lexical entries, syntactic rules or any other grammatical statement. An HPSG theory is simply a set of descriptions; some of those descriptions constrain phrases, while others describe words. In this framework, the lexicon can be thought of as a disjunctive constraint on objects of a certain sort, usually the sort *word*. But any number of principles can be specified in the theory to state generalizations about word objects. As a result, the lexical entries comprising the lexicon as part of the disjunctive constraint on words are less specific and have lost their unique position in specifying lexical information. This suggests that the concept of a lexicon and lexical rules as outside of the theory in the formal sense of King (1989, 1994) are redundant in that one should be able to provide an interpretation of lexical rules on a par with other generalizations in the theory, i.e., as a relation on word objects.

Of course, this does not mean that lexical rules as they existed before, cannot play any role in current HPSG. Rather, it shows that lexical rules as specified by the linguist can be interpreted in two ways-as meta-descriptions relating *lexical entries* (Calcagno 1995), or as descriptions relating *word objects* (Meurers 1995). To distinguish the two approaches in the discussion, a lexical rule under the former approach is called a *Meta-level Lexical Rule* (MLR), while a lexical rule in the latter setup is referred to as a *Description-Level Lexical Rule* (DLR).

3.1. *Defining the basic lexicon*

Corresponding to the two conceptions of a grammar introduced above, there are two options for integrating the lexicon into the HPSG architecture. The first integrates the lexicon as external to the theory and forms the basis of the MLR approach to lexical rules, whereas the second defines the lexicon as part of the theory as needed for the DLR formalization.

3.1.1. *The lexicon as a set external to the theory*

In a traditional perspective distinguishing a lexicon from other grammatical constraints, the natural move is to extend the notion of an HPSG grammar by introducing the *lexicon* as an extra set of descriptions of word objects. A *lexical entry* then is an element of this set.[12] More formally, under this view, a grammar is a triple $G = <\Sigma, \Theta, L>$, with Σ a signature (declaring the linguistic ontology), Θ a theory (a set of descriptions that has to be true of every grammatical object), and L a lexicon (a set of descriptions of objects of type *word*). The denotation of a grammar, then, is the denotation of Θ with the additional restriction that those elements that are of type *word* also have to satisfy (at least) one lexical entry. The denotation of a grammar thus is a subset of the denotation of its theory.

3.1.2. *The lexicon as part of the theory*

The second possibility for expressing a *lexicon* in the HPSG architecture is to include it in the theory as an ordinary implicational constraint on word objects (Meurers 1994:25; Höhle 1996a) like the one shown in Fig. 6.

$$word \rightarrow D_1 \vee D_2 \vee \ldots \vee D_n$$

Fig. 6. The lexicon defined as part of the theory.

A constraint of this form is sometimes called the *Word Principle*, with each D_i ($1 \leq i \leq n, n$ finite) a *lexical entry*, i.e., a description of word objects. Unlike in the first setup, in the Word Principle approach no extension of the notion of an HPSG theory and its interpretation is required. The lexicon is a constraint like all other constraints in the theory and is interpreted in the standard way.

An interesting formal point to note about the Word Principle is that since the length of a description in SRL, just as in standard first-order logic, is required to be finite, the word principle formalization restricts us to a finite set of lexical entries. It is possible to license an infinite number of *word* objects, though, since in principle any description can have an infinite denotation.

Höhle (1996b) remarks that Pollard and Sag (1994:395, fn. 1) conceive the basic lexicon to be an "*exclusive* disjunction of descriptions". This implies a complication of the word principle of Fig. 6 in order to make all disjuncts exclusive, for example as shown in Fig. 7.

$$
\begin{aligned}
\mathit{word} \rightarrow \quad & (D_1 \wedge \neg D_2 \wedge \neg D_3 \wedge \ldots \wedge \neg D_n) \\
& \vee (D_2 \wedge \neg D_1 \wedge \neg D_3 \wedge \ldots \wedge \neg D_n) \\
& \vdots \\
& \vee (D_n \wedge \neg D_1 \wedge \neg D_2 \wedge \ldots \wedge \neg D_{n-1})
\end{aligned}
$$

Fig. 7. Complicating the lexicon to obtain exclusive disjunctions.

It has, however, never been argued why every word should only be described by exactly one disjunct. Furthermore, checking whether a specific word is licensed by a lexicon in such a setup would require considering all descriptions D or the negation thereof – a highly complex task which is virtually impossible for any larger lexicon. We therefore follow Höhle (1996b) in considering such a complication of the word principle to be unjustified.

3.2. *Extending the lexicon with lexical rules*

Now that we have a formal characterization of a basic lexicon, we can turn to the issue of extending this lexicon with lexical rules. We start with lexical rules under the MLR approach before showing how lexical rules as DLRs can be integrated into the theory.

3.2.1. *Extending the lexicon with MLRs*

An MLR is a binary relation between descriptions, which for any description in the domain of the relation (the input entry) will produce a set of descriptions (the output entries). MLRs expand a finite base lexicon by licensing additional lexical entries much in the same way that meta-rules in GPSG (Gazdar et al. 1985) were thought of as expanding a basic set of phrase structure rules by licensing additional phrase structure rules.

Calcagno and Pollard (1995) provide the following definition which uses a least fixed point construction to define a full lexicon on the basis of a base lexicon and a set of lexical rules:

DEFINITION 1 (Full Lexicon under MLR approach). *We assume a finite set $R = \{r_1, \ldots, r_k\}$ of binary relations between formulas, called lexical rules, and a finite set of base lexical entries $B = \{\beta_1, \ldots, \beta_l\}$. Then the full set of lexical entries is the least set L such that:*

- *$B \subset L$; and*
- *for all $\lambda \in L$ and $r \in R$ such that $r(\lambda, \phi)$, $\phi \in L$.*

The full lexicon is defined as the relational closure of the base lexicon under the set of lexical rules. The base lexical entries in the set B are specified by the linguist; the full set of lexical entries is obtained by adding each description to the lexicon set which is related to a base or an already derived lexical entry via one of the lexical rule relations r.

Some consequences of an MLR formalization. As long as the closure under lexical rule application, the full lexicon set L in definition 1 is finite, it is possible to formally express the lexicon either as a distinguished set of descriptions of word objects or as disjuncts on the right-hand side of a Word Principle. The mentioned meta-rules of GPSG were in fact restricted so that only a finite number of phrase structure rules were produced. However, restricting lexical rule application in this way appears to be empirically inadequate or at least in contradiction to the development of HPSG: most current HPSG analyses of Dutch, German, Italian, and French make use of infinite lexica. This is, for example, the case for all proposals working with verbal lexical entries which raise the arguments of a verbal complement in the style of Hinrichs and Nakazawa (1989) that also use lexical rules such as the Complement Extraction Lexical Rule (Pollard & Sag 1994) or the Complement Cliticization Lexical Rule (Miller & Sag 1993, Monachesi 1999) to operate on those raised elements. Also an analysis treating adjunct extraction via lexical rules (Van Noord & Bouma 1994) results in an infinite lexicon. Finally, Carpenter (1991) provides examples from the English verbal system for which recursive rule application and hence a potentially infinite lexicon seems necessary.

Let us illustrate one of these examples in which an infinite number of lexical entries (not the words described) arises in an MLR setup: the interaction of argument raising with the Complement Extraction Lexical Rule. In Fig. 8 we see the essential aspect of the lexical entry for the German auxiliary *haben*, namely the argument raising specification introduced by Hinrichs and Nakazawa (1989) and used in most current HPSG analyses of Germanic and Romance languages. The idea behind the argument raising specification is that such verbs are supposed to combine in a head cluster with the head of their verbal complement. Essentially incorporating the idea of functional composition from categorial grammar (Geach 1970), the unrealized arguments of the selected complement are taken over by the selecting head.

$$
\begin{bmatrix}
\textsc{phon} & \langle \textit{haben} \rangle \\
\textsc{comps} & \left\langle \begin{bmatrix} \textsc{head} & \begin{bmatrix} \textit{verb} \\ \textsc{vform} & \textit{psp} \end{bmatrix} \\ \textsc{comps} & \boxed{1} \end{bmatrix} \middle| \boxed{1} \right\rangle
\end{bmatrix}
$$

Fig. 8. Argument raising in the lexical entry of a German auxiliary.

In Fig. 8, one of the complements which this perfect auxiliary *haben* subcategorizes for is its past participle verbal complement. The rest of the COMPS list (after the | operator in the figure) is specified to be identical to the list of complements which are subcategorized for by the verbal complement. The exact number of complements thus is not fixed in the lexical entry. Note that this property is not just an artefact of ignoring that somewhere down the line of even the longest chain of auxiliaries in a sentence, there will be a full verb serving as verbal complement which has a fixed number of complements. Certain argument raising verbs subcategorize for a nominal object in addition to the verbal complement, in particular the so-called *Accusativum-cum-Infinitivum (AcI)* verbs such as *see, hear*, or *let*. Since AcI verbs can embed each other, regarding the generative potential of the language there thus is no upper limit on the number of complements subcategorized for by a verb.

The Complement Extraction Lexical Rule (CELR) as provided by Pollard and Sag (1994:378) is shown in Fig. 9.[13] The essential effect of the rule is that it removes the element tagged $\boxed{3}$ from the COMPS list of the input in order for this element to be realized non-locally, e.g., as a topicalized constituent. The output of the lexical rule thus has one less element on the COMPS list and can again serve as input to the CELR. The question we are interested in is: How many lexical entries result from the application of the CELR to the entry of the auxiliary we saw in Fig. 8? Given that we showed above that the length of the COMPS list of an entry is not fixed, at least when argument raising verbs are included in a grammar, the answer has to be that the CELR under the MLR perspective produces an infinite number of lexical entries when applied to the lexical entry of an argument raising verb.

$$\begin{bmatrix} \text{ARG-ST} & \langle \ldots \boxed{3} \ldots \rangle \\ \text{COMPS} & \langle \ldots \boxed{3}[\text{LOC } \boxed{1}] \ldots \rangle \\ \text{INHER|SLASH} & \boxed{2} \end{bmatrix} \mapsto \begin{bmatrix} \text{ARG-ST} & \langle \ldots \boxed{4}[\text{LOC } \boxed{1} \ \text{INHER|SLASH} \{\boxed{1}\}] \ldots \rangle \\ \text{COMPS} & \langle \ldots\ldots \rangle \\ \text{INHER|SLASH} & \{\boxed{1}\} \cup \boxed{2} \end{bmatrix}$$

Fig. 9. The Complement Extraction Lexical Rule (Pollard & Sag 1994).

A consequence of such theories licensing infinite lexica is that it commits the MLR approach to a view of the lexicon as a set outside of the theory. This is the case since in SRL it is not possible to specify an infinite disjunction as a description.[14]

Another important consequence of the MLR approach arises from the fact that it is undecidable whether a description is grammatical with respect to an HPSG theory. In the MLR setup it therefore is not possible to restrict the input of lexical rules to those lexical entries describing only grammatical word objects, i.e., words which satisfy the principles expressed in the theory. Adding a test for grammaticality to definition 1 would amount to adding an undecidable precondition to grammar denotation. Expressed differently, the consequence of this is that in an MLR setup the lexical entries in the basic lexicon set are the only part of the grammar that constrains the possible inputs of a lexical rule – it is not possible to require other principles to hold of the inputs to lexical rules, be it to restrict what can constitute a possible base lexical entry or a possible "intermediate" entry, i.e., an entry which is the output of one lexical rule and the input of another one.

A related consequence develops from the fact that not only the lexical entries but also the lexical rules in an MLR approach are introduced as entities separate from the rest of the linguistic theory. It therefore is not possible to use the existing architecture, i.e., principles in the theory, to express generalizations over possible lexical rules. Similarly, there are no mechanisms for encoding a hierarchical organization of lexical rules to organize them in classes with common properties.

3.2.2. *Introducing DLRs into the theory*

A DLR is a binary relation between word objects. While this departs from the more traditional view, in which lexical rules are formalized as meta-relations, it makes it possible to integrate lexical rules at the level at which the other grammatical constraints in the HPSG architecture are expressed. An SRL description denotes a set of objects so that a formula describing both an object and the value of one of its appropriate attributes can be thought of as relating two objects. In the grammar defined in Pollard and Sag (1994), for example, a description of a *functional* head object and its SPEC value expresses such a binary relation holding between a head object and its SPEC value.

Perhaps the simplest way to formalize lexical rules as part of the description language would be to introduce two subtypes of *word*, say *simple-word* and *derived-word* and give *derived-word* an additional appropriate attribute IN with *word* as appropriate value. Fig. 10 shows the relevant portion of the signature. The implicational constraints in Fig. 11 then define the lexicon including lexical rules.

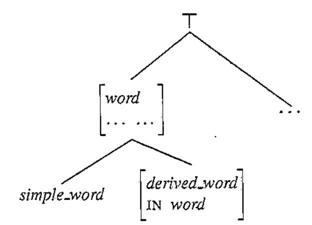


Fig. 10. The signature for the word-in-word encoding.

$$\textit{simple-word} \rightarrow L_1 \vee \ldots \vee L_n$$

$$\textit{derived-word} \rightarrow \left(\left[\text{IN } D_1\right] \wedge E_1\right) \vee \ldots \vee \left(\left[\text{IN } D_m\right] \wedge E_m\right)$$

Fig. 11. The theory for the word-in-word encoding.

In this encoding, the in-description D_j of a DLR $j\,(1 \leq j \leq m)$ is specified on the IN-attribute, while the out-description E_j is specified directly on the *derived-word*.

The disadvantage of this encoding, however, is that if a specific linguistic theory introduces subtypes of *word*, "parallel" subtypes will have to be introduced for *derived-word*. Furthermore, to refer to the output of a lexical rule when we discuss and define the interpretation of lexical rule specifications below, one always has to distinguish between the special attribute IN of words and all its other attributes. To avoid these problems we propose a more modular encoding which clearly separates the lexical rules from the words. Fig. 12 shows an implicational constraint on *word* defining an extended lexicon including lexical rules.

$$\textit{word} \rightarrow \left(L_1 \wedge \left[\text{STORE } \langle\rangle\right]\right) \vee \ldots \vee \left(L_n \wedge \left[\text{STORE } \langle\rangle\right]\right) \vee \boxed{1}\left[\text{STORE } \left\langle \begin{bmatrix} \textit{lex\_rule} \\ \text{OUT } \boxed{1} \end{bmatrix} \right\rangle\right]$$

Fig. 12. A Word Principle for an extended lexicon.

The type *word* is assumed to have an additional appropriate feature STORE, which is *list* valued. Furthermore, a new type *lex_rule* is introduced, having IN and OUT as appropriate features with *word* values. The relevant part of the signature is shown in Fig. 13. The different lexical rules are specified in a constraint on *lex_rule* like the one shown in Fig. 14.

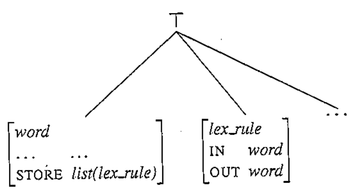


Fig. 13. A signature for the modular lexical rule encoding.

$$\textit{lex\_rule} \rightarrow \begin{bmatrix} \textit{lex\_rule} \\ \text{IN } D_1 \\ \text{OUT } E_1 \end{bmatrix} \vee \ldots \vee \begin{bmatrix} \textit{lex\_rule} \\ \text{IN } D_m \\ \text{OUT } E_m \end{bmatrix}$$

Fig. 14. Defining lexical rule objects in the modular encoding.

Each disjunct on the right-hand side of the implication encodes a lexical rule. We will refer to each such disjunct as a description-level lexical

rule (DLR) and to the in- and out-descriptions D_j and $E_j\,(1 \leq j \leq m)$ as DLR-In and DLR-Out.

So how does this encoding work? The constraint in Fig. 12 says that every object of type *word* is either described by a base lexical entry $L_i\,(1 \leq i \leq n)$ or it is the value of the OUT attribute of a *lex_rule* object. The implicational constraint on *lex_rule* ensures that only a certain set of words are possible values of its OUT attribute, namely those which satisfy one of the out-descriptions E_j in the consequent. The corresponding D_j also has to be consistent and, since the appropriateness conditions for *lex_rule* ensure that the value of an IN feature is of type *word*, it also has to satisfy the constraint on *word*, i.e., one of the lexical entries of Fig. 12. Naturally the lexical entry satisfied can again be the last disjunct, i.e., the output of a lexical rule. Even though the disjunction is finite, we therefore still can license an infinite number of non-isomorphic grammatical word objects via the last disjunct in the Word Principle of Fig. 12.

Finally, we turn to a somewhat different alternative for expressing lexical rules as a binary relation on word objects. This alternative consists of expressing relations by constructs which are part of the relational extension of the description language. This would formalize lexical rules parallel to relations like *append*, or more accurately a binary relation like *member*. If we chose a formal language for HPSG which allows us to use definite relations within the description language, such as the system defined in Götz (2000) which extends King (1989) with ideas from Höhfeld and Smolka (1988) and Dörre (1994), it is possible to represent a lexicon including lexical rules in the formal language without extending the signature. The Figs. 15 and 16 illustrate this possibility.

$$\textit{word} \rightarrow L_1 \vee \ldots \vee L_n \vee \textit{lex\_rule}(\textit{word})$$

Fig. 15. A lexicon with added lexical rule relations.

$$\begin{aligned} \textit{lex\_rule}(D_1) &:= E_1. \\ &\vdots \\ \textit{lex\_rule}(D_m) &:= E_m. \end{aligned}$$

Fig. 16. Defining the lexical rule relation.

Note that a functional notation for relations is used. Just as before, D_j is the in-description of lexical rule j and E_j its out-description. What is different in this encoding is that now the lexical rules are defined on a different level than the *word* objects. As a result, the linguistic ontology does not have to be complicated by book-keeping features like STORE or special types like *lex_rule*. Which *word* objects satisfy our theory is defined using the description language, while the *lex_rule* relation is defined using the relational extension of that description language.

Some consequences of a DLR formalization. Before turning to the consequences of the DLR formalization, we need to pick one of the possibilities discussed above for introducing DLRs into the theory. Since a discussion of how a formal language for HPSG can be extended with relations and which extension is the most appropriate one is a highly complex topic on its own (see, for example, Götz 2000, Richter 1997, 1999, Richter et al. 1999), we avoid this largely orthogonal issue by basing the formalization of DLRs in section 5 on the modular encoding with the *lex_rule* type. Note that the use of STORE and the *lex_rule* type in the modular encoding is quite traditional in that it is an instance of the so-called junk-slot encoding of relations as introduced by Aït-Kaci (1984) and employed by King (1992) and Carpenter (1992).

The key motivation for formalizing lexical rules in HPSG as DLRs develops from the already mentioned fact that in the formal language for HPSG of King (1989, 1994) the notion of an HPSG grammar has been simplified to a point where, from a formal point of view, no distinction is made between lexical entries, syntactic rules or any other grammatical statement. This simple, uniform notion of an HPSG grammar can be maintained if one introduces lexical rules on a par with the other grammatical constraints, i.e., as a description-level mechanism like the DLR encoding described above. Such a tight integration of the lexicon with the rest of the theory is also supported by linguistic phenomena such as idioms, which exhibit a wide range of properties, from purely lexical to productively syntactic.

A formalization of lexical rules as part of the theory differs, however, from a more traditional view of the lexicon where lexical entries are defined in a separate lexicon set and lexical rules as relationships between lexical entries (and not the words described by the entries). We therefore need to investigate, whether it is conceptually sensible to consider DLRs as a formalization of lexical rules in the HPSG framework. That is, apart from being able to express the same generalizations, the important conceptual question is whether properties which were claimed to distinguish lexical rules from other mechanisms, in particular syntactic transformations, still hold for lexical rules in their reincarnation as DLRs.

Lexical vs. structural information and mechanisms. Höhle (1978: 9 ff.) discusses the differences between lexical rules and syntactic transformations based on the setup of a grammar along the lines of Chomsky (1965) (henceforth: ATS). Syntactic transformations operate on (representations of) sentences, which are lexically fully specified. All words in the sentence which the transformation does not explicitly change thus have to occur in the same form in the output. Lexical rules on the other hand, operate on single lexical entries. Words occurring in the syntactic environment of a word licensed by a lexical entry which is the input of a lexical rule thus do not stand in a direct relationship to the words which occur in the syntactic environment of a word licensed by the output of a lexical rule.

In the HPSG architecture of Pollard and Sag (1994), the lexical and the syntactic level of explanation are more difficult to separate. Syntactic structure transformations have never been proposed in this architecture, so that a direct comparison between a syntactic and a lexical mechanism within HPSG is not possible. But one can investigate how a DLR incarnation of lexical rules in HPSG is situated with respect to the classification into lexical and syntactic mechanisms of Höhle (1978).

One relevant difference between ATS and HPSG concerns the status of lexical specification. Contrary to ATS, a word in HPSG has an explicit internal structure, which among other things includes the word's valence requirements. Each valence requirement is a description of those elements which the word must combine with. When a word occurs as the head of an utterance, the valence requirements of a word are identified with the realized arguments.[15] Following Pollard and Sag (1994) most HPSG proposals assume that not the entire information about the realized argument, the *sign*, but only the *synsem* part of an argument is represented in the valence requirements of a word and identified with the arguments realized in an utterance. Properties of signs which are not part of *synsem* are therefore not accessible by looking at the valence requirements as part of the lexical representation of a word: The particular phonological and morphological realization of the arguments (as encoded under the PHON and similar attributes of signs), the information whether an argument is realized as a *word* or a *phrase*, or the constituent structure of the argument in case it is realized as a phrase.[16]

Summing up, this means that in the HPSG setup of Pollard and Sag (1994) one has a clear separation of lexical and syntactic information loci in the sense that a word does not contain information on whether and how its arguments are syntactically realized.[17] Formalizing lexical rules in such an architecture therefore provides us with a lexical mechanism in which – parallel to the characterization of lexical rules by Höhle (1978) mentioned above – words occurring in the syntactic environment of a word which is the input of a lexical rule do not stand in a direct relationship to the words which occur in the syntactic environment of a word licensed by the output of a lexical rule.

The problematic status of the input to lexical rules. After establishing that a formalization of lexical rules as DLRs in a traditional HPSG architecture remains a truly lexical mechanism, we can turn to another issue which in Calcagno and Pollard (1995) is claimed to be problematic for a DLR formalization of lexical rules, the status of the input to lexical rules.

Calcagno and Pollard (1995:6) base their discussion on a word-in-word encoding like the one we introduced in Fig. 11, using SOURCE as attribute name instead of IN, and they point out that this encoding can be equated to that of a unary phrasal schema. The argumentation then runs as follows:

> But this [lexical rule encoding] is problematic. To see why, let's suppose we have a token of a grammatical agentless-passive English sentence such as (1).
>
> (1) Carthage was destroyed.
>
> If the passive lexical rule is indeed a unary schema as we are supposing, then the passive verb *destroyed* must have as its SOURCE some token of the active verb *destroy*. Now consider the SUBJ value of that active verb. It must be a grammatical *synsem* object. But which one? For example, the category of this *synsem* object might be some form of NP, or it might be a *that*-S. If it is an NP, then what kind of NP is it? A pronoun? An anaphor? A nonpronoun? And if it is a *that*-S, what species of state-of-affairs occurs in its CONTENT: *run? sneeze? vibrate*? Of course there is no reasonable answer to these questions; or to put it another way, all answers are equally reasonable. The conclusion that is forced upon us is that the sentence in (1) is infinitely structurally ambiguous, depending on the detailed instantiation of the subject of the active verb. This *reductio ad absurdum* forces us to reject the view of lexical rules as unary schemata.

The basis of this argumentation is of course correct: The passive verbal word is related by the lexical rule to an active verbal word. For the passive verbal word to be grammatical, every substructure of the word, such as the active verbal word which is housed under its SOURCE attribute, also has to be grammatical.

The misconception in the argumentation creeps in from the focus on a particular *token* of a grammatical agentless-passive sentence. To see what is involved here, let us take a step back and consider an ordinary lexical entry like that for the base form verb *laugh* shown in Fig. 17.

$$\left[\begin{array}{ll} \text{PHON} & \langle \textit{laugh} \rangle \\ \text{SYNSEM} & \left[\begin{array}{ll} \text{CAT} & \left[\begin{array}{ll} \text{HEAD} & \left[\begin{array}{l} \textit{verb} \\ \text{VFORM } \textit{bse} \end{array}\right] \\ \text{VAL} & \left[\begin{array}{ll} \text{SUBJ} & \langle \text{NP}_{\boxed{1}} \rangle \\ \text{SPR} & \langle \rangle \\ \text{COMPS} & \langle \rangle \end{array}\right] \end{array}\right] \\ \text{CONT} & \left[\begin{array}{l} \textit{laugh'} \\ \text{ARG1 } \boxed{1} \end{array}\right] \end{array}\right] \end{array}\right]$$

Fig. 17. A lexical entry for *laugh*.

This description will license an untold number of word objects. These tokens will all have the phonology <*laugh*>, the head value *verb*, and bear all other specifications required by the entry. But since objects are total representations, they will also include values for all of the other appropriate attributes and paths. Assuming a traditional HPSG signature, for our example this means that some of the objects described by the lexical entry in Fig. 17 will have a SUBJ value with nominative case, others with accusative case, etc. For some this subject will be a pronominal, for others a non-pronoun. Some of the word objects will have an empty set as CONTEXT|BACKGROUND value while others have a set with, for example, four elements. And so on. Note that this is not just the question of a lexical entry describing words that can be used in different syntactic configurations. The same syntactic configuration can be used in different utterance situations which regarding their grammaticality are not distinguished by the grammar. For example, when the sentence *I laugh* is uttered by me, the CONTEXT|BACKGROUND|SPEAKER index of that sentence will refer to a different person than when the same words are uttered by someone else.

This situation appears to be exactly parallel to the one described as a problem of the DLR approach by Calcagno and Pollard (1995). But is it really? Take the issue of the case of the subject. It cannot be fixed in the lexical entry of *laugh*, since in the utterance *I see him laugh*, the subject bears accusative case, whereas in the occurrence of *laugh* in *I laugh*, it bears nominative case. At this point, one could argue that the problem only arises if one looks at single words instead of complete sentences. In a complete phrase there will be a subject, so that the case value of the subject is fixed. This will, however, not work as a general solution: On the one hand, one would presumably want a linguistic theory to function properly for grammatical signs in general and not only for fully saturated, sentential phrases. On the other hand, even in a fully saturated, sentential phrase not all of the paths are required to have a specific value by the grammar. Returning to the case of the subject of the word *laugh*, consider the utterance *I try to laugh*. Even though it is a fully saturated sentential phrase, the subject of laugh is not (overtly) realized. The control relation between the subject *I* of *try* in the standard HPSG analysis (Pollard & Sag 1994: ch. 3) is established by coindexing, i.e., by specifying in the lexical entry of *try* that the semantic index of the subject of *try* is token identical with that of the subject of *laugh*. As a result, the other attributes of the subject of *laugh* are not fixed by a particular overt syntactic realization and therefore to a large degree arbitrary. As before, one could remedy part of the situation by being more explicit in the specification of the lexical entry of *try*. For example, as shown by Höhle (1983: ch. 6) there are empirical reasons for assuming that the case value of controlled subjects in German is nominative. It is unclear, though, whether one can find similarly well-founded reasons for fixing every attribute value of unrealized controlled subjects. Rather than fixing grammatical attribute values which happen to be unobservable with stipulated values, it seems to be preferable to permit these values to vary freely, i.e., to not use the grammar to distinguish between them if we do not have grammatical evidence for doing so. That a grammar has to permit the values of certain attributes to vary freely becomes particularly clear for features like the CONTEXT value already mentioned above, the value of which is dependent on the particular utterance context and thus cannot be fixed by the grammar alone.

In sum, the formal setup of HPSG is such that for every grammatical token there can be an untold number of other grammatical tokens which differ only with respect to attribute values not distinguished by the theory. Not only is this a consequence of the setup, this state of affairs is actually intended, since it uses the grammar only for its task of singling out the classes of grammatical objects. If certain values of attributes are irrelevant of the grammaticality of a sentence, different uses of this sentence in actual utterances should be allowed to differ with respect to these attribute values. Moreover, there will also be an untold number of exactly identical tokens, which is needed in order to distinguish between accidental identity of objects and identity of objects required by path equalities.

Let us now return to the discussion of the lexical rule example provided by Calcagno and Pollard (1995). By expressing a lexical rule relating a passive verb to an active one, one relates the occurrence of *destroy* in (1) not to a particular instance of the active *destroy*, but to all the instances which allow the lexical rule application. These active instances of the word *destroy* will include some which cannot construct as a daughter in any phrase, it will include some which specify their subject to match a *that*-S argument, and any other possible occurrence of the active verb in any possible utterance.[18]

The interesting status of the input to lexical rules. So far, we have only discussed the potential problems which were argued to arise from the

fact that under a DLR formalization of lexical rules, the word which is the input of the lexical rule also has to be grammatical, i.e., be licensed by a lexical entry and satisfy the other grammatical principles. Turning the coin around, the positive side is that under a DLR formalization we can be sure that only those words which satisfy the theory can be the input of a lexical rule. We believe that this property is of central importance since this property makes it possible to express generalizations over the entities which are lexical rule inputs. If this were not possible, the lexical entry would be the only locus of information which is input to a lexical rule. As a result, all information would have to be repeated in each and every lexical entry, even though – as a whole industry on this topic shows – a significant amount of lexical information is identical for the members of different lexical classes. Not checking the input of a lexical rule for grammaticality would either render lexical rules useless or ban all work on vertical lexical generalizations to outside the currently available formal setup for HPSG.

Let us illustrate the interaction of vertical lexical generalizations and lexical rules with an example taken from the approach to partial fronting phenomena presented in De Kuthy and Meurers (2001). At the heart of the proposal is a lexical principle (i.e., an implicational statement expressing a vertical generalization) which introduces argument raising as a general option for non-finite verbal words. The basic version of this argument raising principle is shown in Fig. 18.

$$\begin{bmatrix} \textit{word} \\ \text{SYNSEM|LOC|CAT|HEAD} \begin{bmatrix} \textit{verb} \\ \text{VFORM} \;\; \textit{bse} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \text{SYNSEM|LOC|CAT|VAL} \begin{bmatrix} \text{SUBJ} & \langle \boxed{1} \rangle \\ \text{COMPS} & \textit{raised}(\boxed{3}) \oplus \boxed{2} \end{bmatrix} \\ \text{ARG-ST} \;\; \langle \boxed{1} \mid \boxed{2} \rangle \wedge \left(\boxed{3} \bigcirc \textit{indep}\right) \end{bmatrix}$$

Fig. 18. The basic lexical argument-raising principle.

This principle applies to base form verbal words and defines how the elements on the argument structure ARG-ST are mapped onto the valence attributes SUBJ and COMPS. The details are not relevant here, but one can note that as part of this mapping, complement requirements of one of the arguments can be raised and added to the COMPS list (via the binary relation *raised*).

As an example for a lexical entry in this setup, consider that of the transitive verb *ausleihen* (*borrow/lend*) in Fig. 19.

$$\begin{bmatrix} \text{PHON} \; \langle \textit{ausleihen} \rangle \\ \text{SYNSEM|LOC|CAT|HEAD} \begin{bmatrix} \textit{verb} \\ \text{VFORM} \;\; \textit{bse} \end{bmatrix} \\ \text{ARG-ST} \left\langle \begin{bmatrix} \text{LOC|CAT|HEAD} \;\; \textit{noun} \end{bmatrix}, \begin{bmatrix} \text{LOC|CAT|HEAD} \;\; \textit{noun} \end{bmatrix} \right\rangle \end{bmatrix}$$

Fig. 19. Lexical entry of a transitive verb.

The base form entry specifies the ARG-ST list, but not the valence attributes SUBJ and COMPS. The principle of Fig. 18 applies to the words described by the entry in Fig. 19 and only those words which also satisfy the consequent of the principle, i.e., identify the valence attribute values with the relevant parts of the argument structure, are grammatical.

Coming to the crucial point of this example, finite verbs are assumed to be derived from their base forms by the lexical rule shown in Fig. 20.

$$\begin{bmatrix} \textit{word} & \\ \textsc{phon} & \boxed{1} \\ \textsc{synsem}|\textsc{loc}|\textsc{cat} & \begin{bmatrix} \textsc{head} & \begin{bmatrix} \textit{verb} & \\ \textsc{vform} & \textit{bse} \end{bmatrix} \\ \textsc{val} & \begin{bmatrix} \textsc{subj} & \boxed{2} \\ \textsc{comps} & \boxed{3} \end{bmatrix} \end{bmatrix} \end{bmatrix} \mapsto \begin{bmatrix} \textsc{phon} & \textit{bse2fin}\left(\boxed{1},\boxed{2}\right) \\ \textsc{synsem}|\textsc{loc}|\textsc{cat} & \begin{bmatrix} \textsc{head}|\textsc{vform} & \textit{fin} \\ \textsc{val} & \begin{bmatrix} \textsc{subj} & \langle\rangle \\ \textsc{comps} & \boxed{2} \oplus \boxed{3} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Fig. 20. A simple finitivization lexical rule.

If the inputs to lexical rules were not checked for grammaticality, this would mean that a base form verb feeding this finitivization lexical rule would not have to be grammatical. The principle of Fig. 18 would therefore not ensure that the valence attributes of the input are identified with the relevant parts of the argument structure so that the valence attributes of the inputs to the lexical rule would be entirely unspecified. As a result, the COMPS list of the finite verbs derived by the lexical rule would be equally unconstrained – which naturally is not the intended result.

Concluding the discussion of this example, we believe it clearly illustrates that a theory including principles generalizing over words only interacts in a reasonable way with lexical rules if the inputs of lexical rules are required to be grammatical.

We already saw in section 3.2.1, though, that under the MLR formalization of lexical rules it is not possible to restrict the input of lexical rules to those entries which describe grammatical words. A meta-level lexical rule therefore can derive grammatical entries from ungrammatical lexical entries as well as from grammatical ones.[19] An MLR formalization of lexical rules therefore cannot be sensibly used for linguistic proposals which include principles generalizing over words, at least when these words are described by lexical entries that can feed a lexical rule.

Carl Pollard and Gosse Bouma (p.c.) mentioned that one might want to exploit the fact that ungrammatical lexical rule input is possible in an MLR setup to encode that such entries obligatorily feed a lexical rule and thereby (possibly) become grammatical. For example, one might want to derive the passive form of the verb *rumour* from the inexistent and therefore supposedly ungrammatical active form.[20] However even

under a description-level formalization, where the input of a lexical rule has to be grammatical, it is possible to express that some word cannot be used unless it has undergone some lexical rule. Such 'phantom' words only have to bear a specification which makes them unusable in any syntactic construction. Ideally, this would be a specification of independently motivated attributes; if this is not possible, a new attribute would have to be introduced for this purpose. In any case, this specification would not keep them from being lexical entries which satisfy the grammatical constraints. In a DLR setup, we can thus exclude these entries from surfacing in phrasal structures without making them ungrammatical.

Expressing constraints on lexical rules. Parallel to the issue of the relationship between the input to a lexical rule and the rest of the linguistic theory, the relationship between the lexical rule itself and the rest of the linguistic theory deserves some attention. The central question here is whether principles in the theory can be used to express generalizations over lexical rules. In the DLR setup, lexical rules are encoded by ordinary linguistic objects. One can therefore restrict the range of possible lexical rules and express generalizations over subclasses of them with the help of ordinary principles. In the MLR setup, lexical rules do not interact with the theory at all; they only serve to extend the lexicon set located outside of the theory. Without extending the setup of HPSG it therefore is not possible to express generalizations over MLRs.

As an example, take the case of an adjuncts-as-complements approach. Ivan Sag (p.c.) suggested that instead of formulating a lexical rule adding the adjuncts onto the complement-list directly (Van Noord & Bouma 1994), or introducing an additional attribute DEPENDENTS to eliminate the lexical rule with a principle adding adjuncts onto this new attribute (Bouma et al. 2001), one could express the addition of adverbials by a constraint on all lexical rules mapping lexemes to words. A sketch of such a constraint in a DLR formalization of lexical rules is shown in Fig. 21.

$$\begin{bmatrix} \textit{lex-rule} & \\ \textsc{in} & \textit{lexeme} \\ \textsc{out} & \textit{word} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{in}|\textsc{arg-st} & \boxed{1} \\ \textsc{out}|\textsc{arg-st} & \boxed{1} \oplus \textit{list}\left(\textit{adverbial}\right) \end{bmatrix}$$

Fig. 21. A constraint on lexical rules.

Working out a proposal along these lines would clearly require more argumentation and most likely result in a more complex version of such a constraint. The simple constraint in Fig. 21 is, however, sufficient to illustrate how the DLR formalization can in principle be used to express constraints on lexical rules in exactly the same way that principles are used to express generalizations over other linguistic objects.

The example illustrates a further property of the DLR approach, namely that it does not have to be restricted to word-to-word mappings. To license lexeme-to-word mappings as assumed in this example, one only has to modify the appropriate value for the attribute IN of type *lex-rule* in the signature shown in Fig. 13 to be a common supertype of *word* and *lexeme*. The advantage of using lexical rules to express such a mapping would be that only those properties which change have to be explicitly included in the lexical rule specification (at least as far as the feature geometry of lexemes corresponds to that of words). The interpretation of the lexical rule specification language introduced in section 4 will make sure that all unchanged properties are carried over.

Principles restricting lexical rules could also be used to express that a more restricted set of lexical rules shares some properties. Since different principles can restrict and interact on different sets and subsets of lexical rule mappings, this allows for a hierarchical organization of constraints on lexical rules. Here hierarchical is intended to mean that when one principle restricts the properties of a set of lexical rules, another principle can restrict further properties of a subset of these lexical rules.

4. A LEXICAL RULE SPECIFICATION LANGUAGE

Having discussed the different possibilities to integrate relations between words or lexical entries into the formal setup of an HPSG grammar, we can now turn to the question how such lexical rule relations can be *specified*. We believe the answer to this question is independent of a particular formal basis for lexical rules. That is, regardless of whether lexical rules relate word objects as in the DLR approach, or lexical entries as in the MLR approach, they are intended to capture the same class of generalizations and a precise language to specify these generalization can be defined independently. To emphasize this point, and to facilitate discussion, we introduce the term *lexical element* as an intentionally neutral term meaning the entities related by a lexical rule.

4.1. *What needs to be expressed?*

So what kind of relation needs to be expressed by a lexical rule? Consider two lexical elements related by a lexical rule. We can distinguish three parts: a) Certain specifications of the input are related to *different* properties of the output. b) Certain specifications of the input are related to *identical* properties of the output. And finally, c), certain specifications of the input have *no relation* to specifications of the output, either because i. the linguist intends those specifications to be unrelated, or ii. because those specifications are appropriate for one lexical element but not the other.

For example, a lexical rule relating German base form verbs to their finite forms, among other things needs to a) relate the base verb form specification and the base morphology to a finite verb form and the corresponding finite morphology, b) ensure that the semantic predicate expressed is the same for both objects, and c-i.) ensure that the finite verb can appear in inverted or non-inverted position regardless of the inversion property of the base verb (which in fact can only occur in non-inverted position). An example for the case c-ii.), where certain properties cannot be transferred, could occur in a nominalization lexical rule which relates verbs to nouns. Since a verb form specification is inappropriate for nouns, that specification cannot be transferred from the verb.

In standard practice, lexical rules in HPSG are written down as two AVMs separated by an arrow, as exemplified by the lexical rule in Fig. 5. At first sight, the AVMs, or more precisely the description language expressions which they stand for, clearly and explicitly express the intended relationship between lexical elements: the AVM to the left of the arrow specifies the domain, while the AVM to the right specifies the range. However, as we will motivate in the following, closer inspection reveals a fundamental unclarity: lexical rules as traditionally specified rely on implicit specifications and the ordinary description language does not allow unambiguous specification of certain relationships. We therefore distinguish the language used by the linguist to write down a lexical rule, the *lexical rule specification language*, from the actual relation intended to be captured. Lexical rule specifications (LRS) are written as “LRS-In $\mapsto$ LRS-Out”. The input- and the output-specification LRS-In and LRS-Out will be specified in an extended version of the description language introduced below, which is designed to provide an unambiguous notation for specifying lexical rule relations.

So in what way is implicit specification used in an LRS? Traditionally, an input to a lexical rule is understood to be minimally altered to obtain an output compatible with LRS-Out: the lexical rule in Fig. 5 “(like all lexical rules in HPSG) preserves all properties of the input not mentioned in the rule.” (Pollard & Sag 1994:314, following Flickinger 1987). Therefore, no specifications expressing identities (i.e., case b discussed above) are included in an LRS. Interpreting the two AVMs as ordinary descriptions would therefore miss part of the intended effect. This idea to preserve properties can be considered an instance of the well-known *frame problem* in Artificial Intelligence (McCarthy & Hayes 1969). We will refer to the additional restrictions on the elements in the range of the rule which are left implicit by the linguist and thus have to be inferred on the basis of the lexical rule specification and the signature as *frame specification* or simply the frame of a lexical rule. The activity of establishing the identities consequently is referred to as *framing*.

The second claim made above was that the standard description language does not allow unambiguous specification of the relationships intended to be expressed. The reason is that no notation is available to distinguish between intended unrelatedness (case c) and mere change of specifications (case a).

4.1.1. *Type specifications and type flattening in LRS-Out*

Take, for example, the signature in Fig. 22, which will serve as the basis for the non-linguistic examples in the following (unless indicated otherwise).

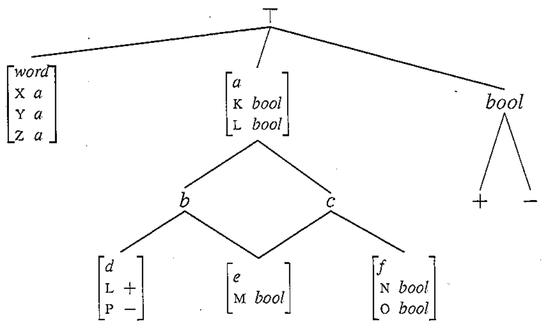


Fig. 22. The signature for the non-linguistic examples.

What kind of X value do we want for the output of the relation specified by the lexical rule specifications in Fig. 23? One possible interpretation is to understand the rule as requiring the output value of X to be *c* if the input's value was incompatible with this assignment, but to keep the value of the input in the output in case it is compatible and more specific.

$$\textit{word} \mapsto \begin{bmatrix}\text{X } c\end{bmatrix}$$

Fig. 23. An example for ambiguous type assignment.

The other possibility is to say that every output of this rule is intended to have *c* as value of X. In other words, the value of X of the output is intended to be unrelated to the value of X in the input. We will refer to this second interpretation as *flattening* of a type assignment.

Since the first, non-flattening interpretation is closer to the intuition of minimally altering the lexical element to obtain an output, we will adopt

this as the standard interpretation of type assignment. To still be able to specify a flattening type assignment, we introduce the new symbol ♭ (flat) and Fig. 24 shows the LRS of Fig. 23 with a flattening type assignment.

$$\textit{word} \mapsto \begin{bmatrix}\text{X } \flat c\end{bmatrix}$$

Fig. 24. A lexical rule using flat.

To get a better feel for the interpretation of the two notations, we take a detailed look at the precise mappings expressed. Fig. 25 illustrates the relation expressed with the LRS of Fig. 23, i.e., without flat.[21]

$$\left\{\langle[\text{X } d],[\text{X } e]\rangle, \langle[\text{X } d],[\text{X } f]\rangle, \langle[\text{X } e],[\text{X } e]\rangle, \langle[\text{X } f],[\text{X } f]\rangle\right\}$$

Fig. 25. The mapping for (non-flattening) type assignment.

Note that it only shows the mapping for the most specific subtypes, the so-called *species* or *varieties*. One obtains the result for a supertype by taking the mapping for each of its most specific subtypes and interpreting the result disjunctively, as illustrated by Fig. 26.

1. $[\text{X } a] \Rightarrow [\text{X } c]$	2. $[\text{X } b] \Rightarrow [\text{X } c]$	3. $[\text{X } c] \Rightarrow [\text{X } c]$
4. $[\text{X } d] \Rightarrow [\text{X } c]$	5. $[\text{X } e] \Rightarrow [\text{X } e]$	6. $[\text{X } f] \Rightarrow [\text{X } f]$

Fig. 26. Lexical entries and what they license via the LRS of Fig. 23.

In the first three cases the value of X of the input is compatible but less specific than the value specified for X in LRS-Out. For these inputs the lexical rule therefore requires the output to have *c* as value for X. The same requirement is made for the output of case four, this time because *d* as value of X of the input is incompatible with the LRS-Out specification. Finally, in cases five and six, the specification of the input is compatible and more specific than the assignment in LRS-Out so that the input's value for X can be carried over to the output.

Taking another look at the second mapping in Fig. 26, one might wonder whether the value of X in the output should not be restricted to *e*, i.e., the common subtype of *b* and *c*, instead of the more general *c* which seems to violate the intuition of minimal alteration. To obtain this interpretation though, we would also have to map *d* into *e* (and not into $f \vee e$), since *b* denotationally is equivalent to $d \vee e$ and we would like to maintain that two denotationally equivalent lexical entries result in equivalent lexical rule outputs. Since restricting the mapping of *d* to *e* in this way is not a sensible interpretation, the mapping of *b* to *e* is undesirable.

Turning to the second lexical rule specification, the LRS with the flattening type assignment shown in Fig. 24, one obtains the mappings in

Fig. 27. Using the flat symbol, the value of X was specified as unrelated to the input and *c* is assigned as value. Therefore every X value is mapped to both species of *c*.

1. $[X\ a] \Rightarrow [X\ c]$	2. $[X\ b] \Rightarrow [X\ c]$	3. $[X\ c] \Rightarrow [X\ c]$
4. $[X\ d] \Rightarrow [X\ c]$	5. $[X\ e] \Rightarrow [X\ c]$	6. $[X\ f] \Rightarrow [X\ c]$

Fig. 27. Applying the flattening type assignment LRS.

The way in which type specifications in an LRS-Out are interpreted is summed up below.

DECISION 1 (Interpretation of type specification in LRS-Out). *A type specification t on path τ in LRS-Out is interpreted as expressing the following relation:*

- *The value of path τ in the output is t' if*
  - *type t' assigned to path τ in the input is a subtype of t, and*
  - *path τ is not specified as flat in LRS-Out*
- *Else, the value of path τ in the output is t.*

Indirect type specifications and normalization. So far, so good; but what about the cases in Fig. 28?

1. $word \mapsto [\text{X}|\text{N}\ +]$
2. $word \mapsto [\text{X}|\text{L}\ -]$
3. $word \mapsto \left[\text{X}\ \begin{bmatrix}\text{K}\ \boxed{1}-\\ \text{L}\ \boxed{1}\end{bmatrix}\right]$

Fig. 28. Lexical rule specifications with implicit type assignments.

The problem is that in those examples even though no type is specified directly in LRS-Out for X, only certain types as values for X in LRS-Out will yield a consistent description. In the first LRS in Fig. 28, the attribute N is only appropriate for objects of type *f*, in the second LRS the attribute L with value "–" is not appropriate for elements of type *d*, and in the third LRS the attribute L is structure shared with K and, since K has "–" as value, again *d* is not a possible value for X.

The solution to this class of problems is to infer all type values of the nodes in LRS-Out which are compatible with the descriptions in LRS-Out and the signature. The task of inferring the compatible species as value of each attribute in a description has already been dealt with: The normalization algorithm of Götz (1994) and Kepser (1994) discussed in detail in section 5 can be used to transform a description into a normal form representation in which (among other things) every attribute is assigned every species consistent with the rest of the description. Based on this normalized representation, the ordinary interpretation for LRS-Out type specifications defined in decision 1 is sufficient.

A related complication can be illustrated with a linguistic example based on the signature of Pollard and Sag (1994). The lexical rule shown in Fig. 29, which we proposed for expository purposes only, licenses predicative versions for all words.

$$word \mapsto [\text{SYNSEM}|\text{LOCAL}|\text{CAT}|\text{HEAD}|\text{PRD}\ +]$$

Fig. 29. PRD-Lexical Rule Specification (for exposition only).

While the PRD value is to be set to +, the usual intention is that the different HEAD types of the input are to be preserved, e.g., if the input of the lexical rule has a HEAD of type *verb*, the output is intended to have a *verb* HEAD as well. As before, normalizing first and then applying the ordinary interpretation for LRS-Out type specifications produces the right result: Normalizing the LRS-Out infers the type *substantive* as HEAD value and by decision 1 a subtype of *substantive*, such as *verb* will be mapped to itself.

Summing up this last part, normalizing LRS-In and LRS-Out allows us to capture rather complex type assignments with the simple interpretation for LRS-Out type specifications defined in decision 1.

Negated type specifications in LRS-Out. Having discussed the effect of positive type specifications in LRS-Out, we can now turn to the interpretation of negated type specifications. It turns out that the SRL setup with its closed world interpretation and a finite set of species allows us to replace all negated type assignments by positive ones. Eliminating negation for negated path equalities does introduce path inequalities, which are dealt with in section 4.1.5 below. But no special treatment of negated type assignments in LRS-Out needs to be defined – the discussion of the positive type assignments above carries over.

Interaction with framing of path equalities. Until now, we have only discussed the effect of type specifications in LRS-Out on typing of the input. So we still need to discuss what effect LRS-Out type specifications are intended to have on the framing of the input's path equalities. There are two possible interpretations here. The first possibility is to argue that a type specification of a path in LRS-Out is a specification of that path and therefore no framing of path equalities takes place. The second possibility is to still ensure framing of those paths even though in some cases this will result in inconsistent outputs.

Consider the LRS and the two possible mappings in Fig. 30.

$$\textit{word} \mapsto \left[\mathrm{X}|\mathrm{K}\ -\right] \qquad \left[\mathrm{X} \left[\begin{array}{ll}\mathrm{K} & \boxed{1} \\ \mathrm{L} & \boxed{1}\end{array}\right]\right] \Rightarrow 1.\left[\mathrm{X} \left[\begin{array}{ll}\mathrm{K} & - \\ \mathrm{L} & \textit{bool}\end{array}\right]\right] \quad 2{:}\left[\mathrm{X} \left[\begin{array}{ll}\mathrm{K} & \boxed{1}- \\ \mathrm{L} & \boxed{1}\end{array}\right]\right]$$

Fig. 30. Type assignment in LRS-Out and path equality in the input.

For the first mapping, the specification of a type for K in LRS-Out is understood as assigning a new value for K and therefore no framing of path equalities takes place. The second case shows the mapping under the second interpretation, where a type specification in LRS does not exclude framing of path equality.

While the second interpretation in this example succeeds in preserving more specifications, this second strategy becomes increasingly complex when looking at more cases. Consider Fig. 31 showing a slightly different mapping with the same LRS but this time with an input with a type specification.

$$\textit{word} \mapsto \left[\mathrm{X}|\mathrm{K}\ -\right] \qquad \left[\mathrm{X} \left[\begin{array}{ll}\mathrm{K} & \boxed{1}+ \\ \mathrm{L} & \boxed{1}\end{array}\right]\right] \Rightarrow 1.\left[\mathrm{X} \left[\begin{array}{ll}\mathrm{K} & - \\ \mathrm{L} & +\end{array}\right]\right] \quad 2.\left[\mathrm{X} \left[\begin{array}{ll}\multicolumn{2}{l}{c} \\ \mathrm{K} & \boxed{1}- \\ \mathrm{L} & \boxed{1}\end{array}\right]\right]$$

Fig. 31. Type assignment in LRS-Out conflicting with path equality plus type assignment in the input.

In this example, the type assignment in LRS-Out conflicts with the path equality and type assignment of the input. Applying the first strategy is straightforward, since the specification of K means that the path equality between K and L holding in the input will not be transferred. To obtain a result for the second strategy, an additional decision needs to be taken which decides how to resolve the conflict between the assignment of L to + and the path equality between K and L. One possibility would be to decide that the specification in LRS-Out always has priority, and that path equalities in the input have priority over type assignments in the input. Such a strategy would then result in the second mapping result shown as part of Fig. 31 above.

However, a very similar conflict can arise where it is not possible to eliminate the input's type specification since it is the appropriate value of the attribute as in the example in Fig. 32.

$$\textit{word} \mapsto \left[\mathrm{X}|\mathrm{K}\ -\right] \qquad \left[\mathrm{X} \left[\begin{array}{ll}\multicolumn{2}{l}{d} \\ \mathrm{K} & \boxed{1}+ \\ \mathrm{L} & \boxed{1}\end{array}\right]\right] \Rightarrow 1.\left[\mathrm{X} \left[\begin{array}{ll}\multicolumn{2}{l}{d} \\ \mathrm{K} & - \\ \mathrm{L} & +\end{array}\right]\right] \quad 2.\ \bot$$

Fig. 32. Type assignment in LRS-Out, path equality and conflicting appropriate types in the input.

The first strategy works as in the previous example: since K is specified in LRS-Out the path equality with K is not transferred to the output. The second strategy assumed for the last example fails this time, since objects of type d allow only + as appropriate value of L. Therefore the conflict between the path equality and the type specification in the input cannot be resolved by eliminating the type specification.

The above discussion shows that the idea to preserve some input path equalities for attributes specified in LRS-Out results in a highly complex problem, basically that of belief revision involved in the task of eliminating a minimal number of facts from a database that has become inconsistent in order to obtain a consistent one. We believe that basing the interpretation of lexical rules on the highly complex strategies needed to successfully deal with such belief revision tasks conflicts with the idea of providing a clear mechanism for expressing horizontal generalizations. We therefore settle for the less expressive but straightforward interpretation which is summed up in decision 2.

DECISION 2 (Framing of path equalities). *Only path equalities holding between paths in the input which are not mentioned in LRS-Out are transferred to the output.*

4.1.2. *Path independence specifications in LRS-Out*

We decided in the previous section that a type or type-flattening specification of a path in LRS-Out prevents framing of a path equality with that path. This brings up the question of how one can make a specification which prevents framing of a path equality for an attribute, without having to specify or flatten its type. For this purpose one could introduce a binary operator $\sharp$ (sharp) to be used to express that no framing of a path equality between the two paths is intended. The notation with different subscripts, i.e., $\sharp_i$, could then be used if multiple pairs of path equalities are to be eliminated. There is a problem with such a notation though, which is illustrated by the LRS in Fig. 33.

$$\textit{word} \mapsto \left[\mathrm{X} \left[\begin{array}{ll}\mathrm{K} & \sharp_2 \\ \mathrm{L} & \sharp_2\end{array}\right]\right]$$

Fig. 33. An LRS with the binary $\sharp$ notation.

The sharps in LRS-Out specify that in the output we don't want to force attributes K and L to be token identical. For certain inputs there are several possibilities for eliminating the path equality restriction on K and L, though, which is shown in Fig. 34.

The problem is that because of the transitivity of path equality, eliminating the path equality between K and L also entails altering the rela-

$$\begin{bmatrix}\text{X} & \begin{bmatrix}\text{K} & \boxed{1}\\ \text{L} & \boxed{1}\\ \text{N} & \boxed{1}\\ \text{O} & \boxed{1}\end{bmatrix}\end{bmatrix} \Rightarrow 1.\begin{bmatrix}\text{X} & \begin{bmatrix}\text{K} & \boxed{1}\\ \text{L} & \boxed{2}\\ \text{N} & \boxed{1}\\ \text{O} & \boxed{2}\end{bmatrix}\end{bmatrix}\quad 2.\begin{bmatrix}\text{X} & \begin{bmatrix}\text{K} & \boxed{2}\\ \text{L} & \boxed{1}\\ \text{N} & \boxed{1}\\ \text{O} & \boxed{2}\end{bmatrix}\end{bmatrix}\quad 3.\begin{bmatrix}\text{X} & \begin{bmatrix}\text{K} & \boxed{1}\\ \text{L} & \boxed{2}\\ \text{N} & \boxed{1}\\ \text{O} & \boxed{1}\end{bmatrix}\end{bmatrix}$$

$$4.\begin{bmatrix}\text{X} & \begin{bmatrix}\text{K} & \boxed{2}\\ \text{L} & \boxed{1}\\ \text{N} & \boxed{1}\\ \text{O} & \boxed{1}\end{bmatrix}\end{bmatrix}\quad 5.\begin{bmatrix}\text{X} & \begin{bmatrix}\text{K} & \boxed{2}\\ \text{L} & \boxed{3}\\ \text{N} & \boxed{1}\\ \text{O} & \boxed{1}\end{bmatrix}\end{bmatrix}$$

Fig. 34. Five possible results.

tionships of K and L to N and Z. To obtain a unique interpretation of the binary $\sharp$ notation one would need to complicate the $\sharp$ notation further. Instead of complicating matters in this way, we introduce path equality elimination $\sharp$ as a unary operator eliminating all path equalities with one path.

To specify an LRS resulting in the five possibilities of Fig. 34, the path equalities which are intended to be kept for an attribute for which $\sharp$ was used to eliminate some path equalities need to be restated. As shown in Fig. 35 it is necessary to repeat certain path equalities because it is only possible to eliminate all path equalities with an attribute and not only those holding between two attributes leaving the others as in the binary notation.

$$1.\ word \mapsto \begin{bmatrix}\text{X} & \begin{bmatrix}\text{K} & \sharp\boxed{2}\\ \text{N} & \sharp\boxed{2}\end{bmatrix}\end{bmatrix}\quad 2.\ word \mapsto \begin{bmatrix}\text{X} & \begin{bmatrix}\text{K} & \sharp\boxed{2}\\ \text{O} & \sharp\boxed{2}\end{bmatrix}\end{bmatrix}\quad 3.\ word \mapsto \begin{bmatrix}\text{X} & \begin{bmatrix}\text{L} & \sharp\end{bmatrix}\end{bmatrix}$$

$$4.\ word \mapsto \begin{bmatrix}\text{X} & \begin{bmatrix}\text{K} & \sharp\end{bmatrix}\end{bmatrix}\quad 5.\ word \mapsto \begin{bmatrix}\text{X} & \begin{bmatrix}\text{K} & \sharp\\ \text{L} & \sharp\end{bmatrix}\end{bmatrix}$$

Fig. 35. LRSs using unary $\sharp$ to specify the results in Fig. 34.

Proceeding to a slightly more complex case, consider the LRS and the example mapping in Fig. 36.

$$word \mapsto \begin{bmatrix}\text{X} & \sharp\end{bmatrix}\qquad 1.\begin{bmatrix}\text{X} & \boxed{1}d\\ \text{Y} & \boxed{1}\\ \text{Z} & \boxed{1}\end{bmatrix} \Rightarrow \begin{bmatrix}\text{X} & d\\ \text{Y} & \boxed{1}d\\ \text{Z} & \boxed{1}\end{bmatrix}\qquad 2.\begin{bmatrix}\text{X} & \boxed{1}\begin{bmatrix}\text{K} & \boxed{2}\\ \text{L} & \boxed{2}\end{bmatrix}\\ \text{Y} & \boxed{1}\\ \text{Z} & \boxed{1}\end{bmatrix} \Rightarrow \begin{bmatrix}\text{X} & \begin{bmatrix}\text{K} & \boxed{2}\\ \text{L} & \boxed{2}\end{bmatrix}\\ \text{Y} & \boxed{1}\begin{bmatrix}\text{K} & \boxed{2}\\ \text{L} & \boxed{2}\end{bmatrix}\\ \text{Z} & \boxed{1}\end{bmatrix}$$

Fig. 36. A more complex example using $\sharp$.

The LRS specifies that the attribute X is independent of the path equalities which held on X in the input. The interesting question is what value is supposed to be assigned to X in the output. Following the intuition that the output should be the minimal alteration of the input required by the specifications in LRS-Out, we interpret the LRS in Fig.

36 to require only that the structure sharing with X is not present in the output. As shown in the first mapping, the type assigned to X is preserved, and the second mapping illustrates that path equalities in a substructure and between substructures is preserved as well. If no framing is intended for the type assignment of X, an additional flat specification has the desired effect. If the attributes of the value of X are intended to also be independent of the input's path equalities, they also have to be specified as sharp. This also highlights that the interaction of sharp and flat specifications is straightforward in that the two do not interact: sharp only has an effect on path equalities whereas flat only effects type values.

4.1.3. *Path equality specifications in LRS-Out*

Turning to the second kind of basic specification in LRS-Out, path equalities, we need to decide on whether any framing is intended for paths specified with a path equality in LRS. This question was already partly answered in section 4.1.1, where we decided to not assume framing of the input's path equalities for attributes which are specified in LRS-Out. The remaining question is whether type values of the input should be transferred to paths in the output, for which a path equality is defined in LRS-Out.

Recall that the motivation for restricting framing of path equalities to unspecified LRS-Out paths came from the insight that highly complex strategies are needed to decide how to resolve a conflict resulting from framing of a path equality in the input and an incompatible type specification to one of the paths in LRS-Out. The situation we are faced with now is a mirror image of this problem: how should one resolve a conflict resulting from framing of a type specification in the input and a path equality in LRS-Out. While it might be possible to develop a strategy to answer this question, it will in all likelihood be equally complex as the answers to the mirror-image problem discussed in section 4.1.1. Rather than engage in this traditional problem, we therefore follow the same strategy as in the earlier section and propose to avoid these conflicts all together by not framing type specifications of the input for paths which occur in a path equality in LRS-Out.

DECISION 3 (Interpretation of path equalities in LRS-Out). *A path equality between two paths τ_1 and τ_2 in LRS-Out is interpreted as preventing framing of the input's type values of τ_1 or τ_2.*

4.1.4. *Specifying identities between LRS-In and LRS-Out*

In a useful lexical rule specification language it must be possible to express that an attribute in the output is supposed to be assigned the value which another attribute has in the input. Traditionally, the notation for specifying structure sharing between two paths of an object has been carried over for this use, as illustrated for example by the use of the tags $\boxed{1}$, $\boxed{2}$, and $\boxed{3}$ in LRS-In and LRS-Out of the passive LRS shown in Fig. 5.

Under a DLR approach, the use of path equalities for this purpose makes sense since DLRs are represented just like other linguist objects. Path equalities between parts of the IN and the OUT attribute of *lex_rule* objects (or whatever DLR representation one chooses) can therefore receive the ordinary structure sharing interpretation. We will therefore not complicate the specification language for DLRs for this purpose.[22]

For the MLR approach, one cannot use ordinary path equality to relate input to output of a lexical rule. Path equality denotes token identity of objects, but an MLR relates descriptions of objects, not the objects themselves. A specification language for MLRs would therefore have to introduce meta-variables for this purpose.

4.1.5. *Path inequality specifications in LRS-Out*

As the final kind of specification that can occur in LRS-Out we now discuss path inequalities. Consider the example shown in the left half of Fig. 37.

$$\textit{word} \mapsto \left[\mathrm{X}|\mathrm{K} \not\approx \mathrm{X}|\mathrm{L}\right] \qquad \begin{bmatrix} \mathrm{X} & \begin{bmatrix} \mathrm{K} & \boxed{1} \\ \mathrm{L} & \boxed{1} \\ \mathrm{N} & \boxed{1} \\ \mathrm{O} & \boxed{1} \end{bmatrix} \end{bmatrix} \Rightarrow \begin{bmatrix} \mathrm{X} & \begin{bmatrix} \mathrm{K} & \textit{bool} \\ \mathrm{L} & \textit{bool} \\ \mathrm{N} & \boxed{1} \\ \mathrm{O} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

Fig. 37. Path inequalities in LRS-Out.

The case of path inequality specifications in LRS-Out is very similar to the case of the binary sharp notation discussed above. The problem is that because of the transitivity of path equality, requiring a path inequality to hold of two paths in the output of a lexical rule cannot be accomplished by adding the path inequality and removing a possibly occurring path equality between those paths. One additionally has to decide what happens with other path equalities in which those two paths occur.

The decision 2 restricting framing of path equalities to paths not mentioned in LRS-Out therefore also appears to make sense for path inequalities specified in LRS-Out. An example mapping for this interpretation is shown in the right half of Fig. 37. Note that in case one does want to keep the path equalities in the output which hold with one of the inequated paths, one can include a meta variable between that attribute in LRS-In and LRS-Out to obtain the desired effect.

4.2. *Is automatic framing reasonable?*

After this long discussion of the specification language, one might wonder whether it is not an artifact of assuming automatic framing that a special specification language is needed. After all, when writing down a lexical rule, the linguist only needs to express two of of the three cases (relating differing properties, relating identical properties, unrelated properties). When the linguist specifies those properties which are intended to differ (a) and one more case (b or c), the third kind can be deduced; i.e., it does not have to be expressed explicitly and could be called the "default" specification of lexical rules (in a non-technical sense).

So there really are two possibilities here, of which we have only pursued one above: We discussed an LRS notation in which we explicitly have to mention those properties which are intended to be unrelated in the elements described by LRS-In and LRS-Out (case c-i.). For this we had to introduce additional notation, but the positive side of this was that no explicit specifications are needed for the case in which specifications are intended to remain unchanged, i.e., automatic framing takes place. The other possibility, however, would be to not have automatic framing and instead have an LRS notation in which those properties which are intended to be identical in the elements described by LRS-In and LRS-Out (case b) are explicitly mentioned. The non-related properties can then remain unexpressed – which eliminates the need for the extra notation introduced above.

At first sight, it does indeed seem natural to ask the linguist to express in an LRS those specifications which relate properties, i.e., cases a) and b), and keep unexpressed which parts of the objects are unrelated (case c). However, in highly lexicalized theories like HPSG, a lexical entry contains many specifications of which only few are relevant in a specific lexical rule. Asking the linguist to explicitly specify that all those specifications without relevance to the lexical rule are identical in the elements related (in case they are appropriate) would thus amount to asking for a lexical rule with many specifications which are of no direct importance to what the lexical rule is intended to do.

Furthermore, as discussed in Meurers (1994: sec. 4.1.3), specifying all identities by hand in many cases can only be achieved by splitting up a lexical rule into several instances. This is the case whenever one needs to specify the type of an element an attribute of which gets specified in the lexical rule output.[23] A second case which requires splitting up the LRS is when one has to specify framing of the value of those attributes which are only appropriate for some of the elements in the domain. Finally, a significant problem can arise from having to explicitly specify framing of the different path equalities which can occur in inputs to a lexical rule.

So, while for simple lexical rules one could specify framing of identical specifications by hand and the ordinary specification language would not have to be extended, for most cases it seems well motivated to assume automatic framing and specify the lexical rules with the extended specification language introduced above.

Summing up, we have argued that additional notation needs to be introduced to obtain a precise specification language. Additional notation is introduced for the case in which non-relatedness is intended, i.e., to mark those linguistic specifications which should not be altered by framing. Two new symbols $\flat$ and $\sharp$ are introduced and $\flat$ is used to mark a type specification as independent of framing, while $\sharp$ marks an attribute as independent of path equality framing.

5. A DLR FORMALIZATION OF THE LEXICAL RULE SPECIFICATION LANGUAGE

After exploring the lexical rule specification in the previous section, we now turn to a particular formalization of this specification language in terms of description-level lexical rules (DLRs), which were introduced in section 3. We start with a review of the formal setup of HPSG on which our approach is based, before turning to the lexical rule related definitions in section 5.2 and an example in section 5.3.

5.1. *A mathematical foundation for HPSG: SRL*

As the formal basis of our approach we assume the logical setup of King (1989) which in King (1994) is shown to provide the foundation desired for HPSG in Pollard and Sag (1994). The formal language defined in the following is a version of the one proposed by King.

5.1.1. *Syntax*

Definition 1 (Signature). *A* **signature** Σ *is a triple* $\langle \mathcal{S}, \mathcal{A}, approp \rangle$ *s.t.*

- $\mathcal{A}$ *is a finite set of* **attribute names**
- $\mathcal{S}$ *is a finite set of* **varieties** *(also called species or most specific types)*[24]
- $approp : \mathcal{S} \times \mathcal{A} \rightarrow Pow(\mathcal{S})$ *is a total function from pairs of varieties and attribute names to sets of varieties*

Everything which follows is done with respect to a signature. For notational convenience we will work with an implicit signature $\langle \mathcal{S}, \mathcal{A}, approp \rangle$. This is possible since at no point in our proposal do we have to alter the signature.

Definition 2 (Term). *Let* $:$ *be a reserved symbol, the root symbol of a path. A* **term** *is a member of the smallest set* $\mathcal{T}$ *s.t.*

- $: \in \mathcal{T}$ *and*
- $\tau\alpha \in \mathcal{T}$ *if* $\tau \in \mathcal{T}$ *and* $\alpha \in \mathcal{A}$

Definition 3 (Description). *Let* $(,), \sim, \approx, \neg, \wedge, \vee$ *and* $\rightarrow$ *be reserved symbols. A* **description** *is a member of the smallest set* $\mathcal{K}$. *s.t.*

- $\tau \sim \phi \in \mathcal{K}$ *if* $\tau \in \mathcal{T}$ *and* $\phi \in \mathcal{S}$
- $\tau_1 \approx \tau_2 \in \mathcal{K}$ *if* $\tau_1, \tau_2 \in \mathcal{T}$
- $\neg\delta \in \mathcal{K}$ *if* $\delta \in \mathcal{K}$
- $(\delta_1 \wedge \delta_2), (\delta_1 \vee \delta_2), (\delta_1 \rightarrow \delta_2) \in \mathcal{K}$ *if* $\delta_1, \delta_2 \in \mathcal{K}$

Definition 4 (Theory). *A* **theory** Θ *is a subset of* $\mathcal{K} (\Theta \subseteq \mathcal{K})$.

Definition 5 (Set of Literals). *A* **set of literals** Σ *is a proper subset of the set of descriptions* $\mathcal{K}$, *i.e.,* $\Sigma \subset \mathcal{K}$, *s.t. each* $\delta \in \Sigma$ *has one of the four forms* $(\tau, \tau_1, \tau_2 \in \mathcal{T}; \phi \in \mathcal{S})$:

- $\tau \sim \phi$
- $\tau_1 \approx \tau_2$
- $\neg\tau \sim \phi$
- $\neg\tau_1 \approx \tau_2$

5.1.2. *Semantics*

Definition 6 (Interpretation of a Signature). *An* **interpretation** I *is a triple* $\langle$U, S, A$\rangle$ *s.t.*

- U *is a* **set of objects**. *the domain of* I,
- S : U $\rightarrow \mathcal{S}$ *is a total function from the set of objects to the set of varieties, the* **variety assignment** *function,*

- $A : \mathcal{A} \to \{U \rightharpoonup U\}$ [25] *is an* **attribute interpretation** *function s.t. for each* $u \in U$ *and* $\alpha \in \mathcal{A}$:

  - *if* $A(\alpha)(u)$ *is defined then* $S(A(\alpha)(u)) \in approp(S(u), \alpha)$, *and*
  - *if* $approp(S(u), \alpha) \neq \emptyset$ *then* $A(\alpha)(u)$ *is defined.*

Definition 7 (Interpretation of Terms). $[\,]^I : \mathcal{T} \to \{U \rightharpoonup U\}$ *is a* **term interpretation** *function over interpretation* $I = \langle U, S, A \rangle$ *s.t.*

- $[:]^I$ *is the identity function on* U, *and*
- $[\tau\alpha]^I$ *is the functional composition of* $[\tau]^I$ *and* $A(\alpha)$.

Definition 8 (Interpretation of Descriptions). $D_I() : \mathcal{K} \to Pow(U)$ *is a* **description interpretation** *function over interpretation* $I = \langle U, S, A \rangle$ *s.t.* $(\tau, \tau_1, \tau_2 \in \mathcal{T}; \phi \in \mathcal{S}; \delta, \delta_1, \delta_2 \in \mathcal{K})$:

- $D_I(\tau \sim \phi) = \{u \in U \mid [\tau]^I(u) \text{ is defined and } S([\tau]^I(u)) = \phi\}$
- $D_I(\tau_1 \approx \tau_2) = \left\{ u \in U \;\middle|\; \begin{array}{l} [\tau_1]^I(u) \text{ is defined,} \\ [\tau_2]^I(u) \text{ is defined, and} \\ [\tau_1]^I(u) = [\tau_2]^I(u) \end{array} \right\}$
- $D_I(\neg\delta) = U \backslash D_I(\delta)$
- $D_I((\delta_1 \wedge \delta_2)) = D_I(\delta_1) \cap D_I(\delta_2)$
- $D_I((\delta_1 \vee \delta_2)) = D_I(\delta_1) \cup D_I(\delta_2)$
- $D_I((\delta_1 \to \delta_2)) = (U \backslash D_I(\delta_1)) \cup D_I(\delta_2)$

Definition 9 (Interpretation of a Theory). *A theory is interpreted conjunctively.* $[\![\,]\!]^I : Pow(\mathcal{K}) \to Pow(U)$ *is a* **theory interpretation** *function over interpretation* $I = \langle U, S, A \rangle$ *s.t.* $[\![\Theta]\!]^I = \cap\{D_I(\delta) | \delta \in \mathcal{K}\}$

Definition 10 (Satisfiability). *A theory* Θ *is* **satisfiable** *iff there is an interpretation* I *s.t.* $[\![\Theta]\!]^I \neq \emptyset$

Definition 11 (Model). *An interpretation* $I = \langle U, S, A \rangle$ *is a* **model** *of a theory* Θ *if* $[\![\Theta]\!]^I = U$

The definitions above define a class of formal languages which can be used to express HPSG grammars. We only list these definitions here to make it possible to follow the formal definition and interpretation of the lexical rule specification language in the next sections. The reader interested in a discussion of the formal language of SRL is referred to King (1994).

5.2. *The lexical rule specification language*

5.2.1. *Syntax*

Definition 12 (Lexical Rule Signature). *Every signature* Σ *for which the following condition holds is a lexical rule signature* Σ_{lr}.

- $lex\_rule \in \mathcal{S}$ *and*
- IN, OUT $\in \mathcal{A}$ *and*
- $approp(lex\_rule, \text{IN}) = \{\text{word}\}$ *and*
- $approp(lex\_rule, \text{OUT}) = \{\text{word}\}$.

Definition 13 (L-Description). *Let* $\flat$ *and* $\sharp$ *be reserved symbols. With respect to a lexical rule signature* Σ_{lr} *let* $\mathcal{T}$ *be a set of terms,* $\mathcal{K}$ *a set of descriptions. A L-description is a member of the smallest set* $\mathcal{K}_\mathcal{I}$ *s.t.*

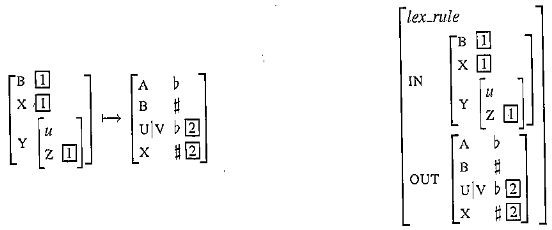
- $d \in \mathcal{K}_\mathcal{I}$ *if* $d \in \mathcal{K}$ *and*
- $:\text{OUT}\mu \sim \flat \in \mathcal{K}_\mathcal{I}$ *if* $\mu \in \mathcal{A}^+$ *and*
- $:\text{OUT}\mu \sim \sharp \in \mathcal{K}_\mathcal{I}$ *if* $\mu \in \mathcal{A}^+$.

Definition 14 (Lexical Rule Specification). *With respect to a given lexical rule signature* Σ_{lr} *a lexical rule specification LRS is a subset of the set of L-descriptions* $\mathcal{K}_\mathcal{I}$ *containing at least the following literals* $(\phi \in \mathcal{S}; \mu, \mu_1, \mu_2 \in \mathcal{A}^+)$:

- $: \sim lex\_rule$ *and*
- $:\text{OUT}\mu \sim \phi$ *or* $:\text{OUT}\mu_1 \approx :\text{OUT}\mu_2$ *or* $:\text{OUT}\mu \sim \flat$ *or* $:\text{OUT}\mu \sim \sharp$.

There's nothing complicated going on here. We just add the additional LRS notation by defining L-formulas with respect to a lexical rule signature. A lexical rule specification then consists of L-formulas, and for convenience sake we ask for an LRS-Out containing at least one specification.

In most HPSG theories proposed in the literature, AVMs are used as descriptions instead of the term notation introduced above. AVMs can be seen as a kind of normal form representation for descriptions. Now that we've introduced the formal lexical rule specification language, let us illustrate the different ways in which one can write down LRSs with an example (which is not intended to say much but just show the way things are written down). We will use the notation shown in figure 38 on the left as shorthand for the AVM shown on the right, which in the formal notation defined above is expressed as shown below that.[26]



$:\sim \textit{lex\_rule} \;\wedge\; \text{:IN B} \approx \text{:IN X} \;\wedge\; \text{:IN X} \approx \text{:IN Y Z} \;\wedge\; \text{:IN Y} \sim u \;\wedge$

$\text{:OUT A} \sim \flat \;\wedge\; \text{:OUT B} \sim \sharp \;\wedge\; \text{:OUT U V} \sim \flat \;\wedge\; \text{:OUT U V} \approx \text{:OUT X} \;\wedge\; \text{:OUT X} \sim \sharp$

Fig. 38. Three ways to write down LRSs.

A normal form for L-descriptions. In section 4.1.1 we saw that the L-formulas making up the LRS need to be normalized to have a consistent variety assigned to each defined attribute, which is needed for the mapping from LRSs to LRs. This section serves to introduce a normal form for descriptions. It reports work carried out in Götz (1994) and Kepser (1994). Originally, the normalization algorithm is used to determine if a given description is satisfiable.

The linguist writes down LRSs. So we want to normalize L-formulas, not simple descriptions. Since L-formulas are a simple extension of descriptions with two additional statements for type and path equality elimination, we only need to make minimal changes to the description normalization algorithm of Götz (1994) to obtain an algorithm which transforms an L-formula into normal form. First, we need to introduce some additional terminology.

Definition 15 (Terms and subterms in Σ). *The set* TERM(Σ) *contains all paths occurring in a set of literals Σ and their subpaths* ($\tau, \tau' \in \mathcal{T}$; $\pi \in \mathcal{A}^*$; $\psi \in \mathcal{S} \cup \{\flat, \sharp\}$):

$$\text{TERM}(\Sigma) = \{\tau \mid (\neg)\tau\pi \approx \tau' \in \Sigma\} \cup \{\tau \mid (\neg)\tau' \approx \tau\pi \in \Sigma\} \cup \{\tau \mid (\neg)\tau\pi \sim \psi \in \Sigma\}$$

Definition 16 (Clause and Matrix)

- *A* **clause** Σ *is a finite (possibly empty) set of literals.*
- *A* **matrix** Γ *is a finite (possibly empty) set of clauses.*

Definition 17 (Interpretation of a Clause and a Matrix)

- *A* **clause** *is interpreted conjunctively.*

  If Σ is a clause, then $D_I(\Sigma) = \cap_{\delta \in \Sigma} D_I(\delta)$.

- *A* **matrix** *is interpreted disjunctively.*

  If Γ is a matrix, then $D_I(\Gamma) = \cup_{\Sigma \in \Gamma} D_I(\Gamma)$.

The conversion from L-formulas to its normal form proceeds in two steps. First, the L-formula is transformed into disjunctive normal form, i.e., where all negations are pulled in and the disjuncts are on the top level. The resulting matrix Γ is a finite set, each element of which represents one disjunct. Each disjunct is a clause which consists of a finite set of literals. Since the transformation into disjunctive normal form is a rather standard procedure, we just assume its existence here. Second, the resulting matrix is normalized. We start with a declarative characterization of what it means for an L-formula to be in normal form.

Definition 18 (Normal Clause). *A set Σ of literals is* **normal** *iff the following conditions hold* ($\tau, \tau_1, \tau_2 \in$ TERM(Σ); $\phi, \phi_1, \phi_2 \in \mathcal{S}$; $\psi \in \mathcal{S} \cup \{\flat, \sharp\}$; $\alpha \in \mathcal{A}$; $\pi \in \mathcal{A}^*$)

1. $: \approx : \in \Sigma$ *(root is defined)*
2. *if* $\tau_1 \approx \tau_2 \in \Sigma$ *then* $\tau_2 \approx \tau_1 \in \Sigma$ *(symmetry of $\approx$)*
3. *if* $\tau_1 \approx \tau_2, \tau_2 \approx \tau_3 \in \Sigma$ *then* $\tau_1 \approx \tau_3 \in \Sigma$ *(transitivity)*
4. *if* $\tau\pi \approx \tau\pi \in \Sigma$ *then* $\tau \approx \tau \in \Sigma$ *(prefix closure)*
5. *if* $\tau_1 \approx \tau_2, \tau_1\pi \approx \tau_1\pi, \tau_2\pi \approx \tau_2\pi \in \Sigma$ *then* $\tau_1\pi \approx \tau_2\pi \in \Sigma$ *($\approx$ and path extensions)*
6. *if* $\tau \approx \tau \in \Sigma$ *then for some* $\phi \in \mathcal{S}, \tau \sim \phi \in \Sigma$ *(exhaustive typing)*
7. *if for some* $\psi \in \mathcal{S} \cup \{\flat, \sharp\}, \tau \sim \psi \in \Sigma$ *then* $\tau \approx \tau \in \Sigma$ *($\sim$ path is defined)*

8. *if* $\tau_1 \approx \tau_2 \in \Sigma, \tau_1 \sim \phi_1 \in \Sigma, \tau_2 \sim \phi_2 \in \Sigma$ *then* $\phi_1 = \phi_2$ *($\approx$ and $\sim$)*

9. *if* $\tau \sim \phi_1 \in \Sigma, \tau\alpha \sim \phi_2 \in \Sigma$ *then* $\phi_2 \in approp(\phi_1, \alpha)$ *(appropriateness 1)*

10. *if* $\tau \sim \phi \in \Sigma$, $\tau\alpha \in \text{TERM}(\Sigma)$, $approp(\phi, \alpha) \neq \emptyset$ *then*
$\tau\alpha \approx \tau\alpha \in \Sigma$ *(appropriateness 2)*

11. *if* $\neg\delta \in \Sigma$ then $\delta \notin \Sigma$ *(no contradictions)*

12. *if* $:\text{OUT}\pi\alpha \approx :\text{OUT}\pi\alpha$, $:\text{IN}\pi \sim \phi \in \Sigma$, $approp(\phi, \alpha) \neq \emptyset$ *then*
$:\text{IN}\pi\alpha \approx :\text{IN}\pi\alpha \in \Sigma$ *(corresponding in-paths are defined)*

The algorithm which takes an L-descriptions as a DNF matrix and returns its normal form is given below as a set of rewrite rules on sets of clauses. Γ is used as variable over sets of clauses and Σ as variable over clauses. Readers interested in the formal properties of the algorithm and a discussion of the normal form are referred to Kepser (1994: sec. II).

ALGORITHM 1 (Clause Normalization). *The algorithm consists of a sequence rewrite rule applications. One step of the algorithm is the application of exactly one rewrite rule. The algorithm terminates if no rule can be applied (any more). A rule applies to a set of clauses Γ' only if the left hand side of the rule matches Γ' and if the right hand side is a valid set description under the same variable assignment. The rewrite rules are* ($\phi_1, \phi_2 \in \mathcal{S}; \psi \in \mathcal{S} \cup \{\flat, \sharp\}; \alpha \in \mathcal{A}; \pi \in \mathcal{A}^*$):

(1) $$\Gamma \uplus \{\Sigma\} \rightarrow \Gamma \cup \{\Sigma \uplus \{:\approx:\}\}$$

(2) $$\Gamma \uplus \{\Sigma \uplus \{\tau_1 \approx \tau_2\}\} \rightarrow \Gamma \cup \left\{ \begin{array}{c} \Sigma \uplus \{\tau_1 \approx \tau_2\} \\ \uplus \{\tau_2 \approx \tau_1\} \end{array} \right\}$$

(3) $$\Gamma \uplus \left\{ \Sigma \uplus \left\{ \begin{array}{c} \tau_1 \approx \tau_2, \\ \tau_2 \approx \tau_3 \end{array} \right\} \right\} \rightarrow \Gamma \cup \left\{ \begin{array}{c} \Sigma \uplus \left\{ \begin{array}{c} \tau_1 \approx \tau_2, \\ \tau_2 \approx \tau_3 \end{array} \right\} \\ \uplus \{\tau_1 \approx \tau_3\} \end{array} \right\}$$

(4) $$\Gamma \uplus \{\Sigma \uplus \{\tau\sigma \approx \tau\sigma\}\} \rightarrow \Gamma \cup \{\Sigma \uplus \{\tau\sigma \approx \tau\sigma, \tau \approx \tau\}\}$$

(5) $$\Gamma \uplus \left\{ \Sigma \uplus \left\{ \begin{array}{c} \tau_1 \approx \tau_2, \\ \tau_1\sigma \approx \tau_1\sigma, \\ \tau_2\sigma \approx \tau_2\sigma \end{array} \right\} \right\} \rightarrow \Gamma \cup \left\{ \begin{array}{c} \Sigma \uplus \left\{ \begin{array}{c} \tau_1 \approx \tau_2, \\ \tau_1\sigma \approx \tau_1\sigma, \\ \tau_2\sigma \approx \tau_2\sigma \end{array} \right\} \\ \uplus \{\tau_1\sigma \approx \tau_2\sigma\} \end{array} \right\}$$

(6) $$\Gamma \uplus \{\Sigma \uplus \{\tau \approx \tau\}\} \rightarrow \Gamma \cup \left\{ \Sigma \uplus \left\{ \begin{array}{c} \tau \approx \tau, \\ \tau \sim \phi \end{array} \right\} \mid \phi \in \mathcal{S} \right\}, \quad \textit{if } \forall \phi'.\tau \sim \phi' \notin \Sigma$$

(7) $$\Gamma \uplus \{\Sigma \uplus \{\tau \sim \psi\}\} \rightarrow \Gamma \cup \{\Sigma \uplus \{\tau \sim \psi, \tau \approx \tau\}\}$$

(8) $$\Gamma \uplus \left\{ \Sigma \uplus \left\{ \begin{array}{c} \tau_1 \approx \tau_2, \\ \tau_1 \sim \phi_1, \\ \tau_2 \sim \phi_2 \end{array} \right\} \right\} \rightarrow \Gamma, \textit{if } \phi_1 \neq \phi_2$$

(9) $$\Gamma \uplus \left\{ \Sigma \uplus \left\{ \begin{array}{c} \tau \sim \phi_1, \\ \tau\alpha \sim \phi_2 \end{array} \right\} \right\} \rightarrow \Gamma, \textit{if } \phi_2 \notin \textit{approp}(\phi_1, \alpha)$$

(10) $$\Gamma \uplus \{\Sigma \uplus \{\tau \sim \phi\}\} \rightarrow \Gamma \cup \{\Sigma \uplus \{\tau \sim \phi, \tau\alpha \approx \tau\alpha\}\}, \quad \textit{if } \tau\alpha \in \text{TERM}(\Sigma) \textit{ and } \textit{approp}(\phi, \alpha) \neq \emptyset$$

(11) $$\Gamma \uplus \{\Sigma \uplus \{\delta, \neg\delta\}\} \rightarrow \Gamma, \textit{for any positive literal } \delta$$

(12) $$\Gamma \uplus \left\{ \begin{array}{l} \Sigma \uplus \\ \left\{ \begin{array}{l} :\text{OUT}\pi\alpha \approx :\text{OUT}\pi\alpha, \\ :\text{IN}\pi \sim \phi \end{array} \right\} \end{array} \right\} \rightarrow \Gamma \cup \left\{ \begin{array}{l} \Sigma \uplus \\ \left\{ \begin{array}{l} :\text{OUT}\pi\alpha \approx :\text{OUT}\pi\alpha, \\ :\text{IN}\pi \sim \phi, \\ :\text{IN}\pi\alpha \approx :\text{IN}\pi\alpha \end{array} \right\} \end{array} \right\}, \quad \textit{if } \textit{approp}(\phi, \alpha) \neq \emptyset$$

Each rewrite rule corresponds to a line in the definition of a normal clause. Line 3 of definition 18, for example, demands transitivity of path equality. The corresponding rewrite rule (3) in algorithm 1 picks out a clause with two literals expressing path equalities and adds a literal expressing the path equality resulting from transitivity, if it is not already part of the clause. Note the use of ordinary ($\cup$) and disjoint union ($\uplus$). The last occurrence of disjoint union in the rewrite rule (3) ensures that this rule will only apply if the literal to be added was not part of the original clause, i.e., if transitivity for the two literals did not already hold in Σ.

The original normalization algorithm of Götz (1994) consists of rules (1)–(11). Since we are dealing with L-descriptions, we additionally have to take care of terms including the new symbols $\flat$ and $\sharp$. To do so, Götz's original rule (6) was modified to also define those paths which bear one of the new specifications. Note that once a path is defined in this way, the rest of the algorithm will ensure that each subpath is also defined and that each (sub)path is assigned the possible varieties.

Finally, rule (12) is an addition to the original algorithm that is specific to lexical rule representations. It ensures that for each path in the out-description the corresponding path in the in-description is introduced, if it is appropriate.

5.2.2. *Semantics*

We define an algorithm which realizes a function from lexical rules as specified by the linguist (LRS) to enriched descriptions of lexical rule objects which can be given the standard set theoretical interpretation defined in section 5.1. The conversion from LRS to ordinary descriptions proceeds in two steps. First, the LRS is converted into normal form, then the normal form LRS is enriched with additional path equalities and variety assignments to encode the framing which is only implicit in the LRS. As a result of enriching the LRS we obtain an ordinary description, i.e., an LR, which is interpreted in the normal way.

Enriching an LRS matrix. We saw in section 5.2.1 what it means for an L-formula to be in normal form. Now we turn to the enriching algorithm.

ALGORITHM 2 (Enriching a normalized lex_rule description). *The input to the enriching algorithm is an LRS in normal form. A normalized LRS is a matrix Γ^{lrs}, a finite set, each element of which represents one disjunct. Each disjunct is a clause which consists of a finite set of literals. The enriching algorithm consists of the following three steps:*

1. *For every clause Σ in the matrix Γ^{lrs}, define a new matrix $\Gamma = \{\Sigma\}$.*

2. *With each such Γ obtain an enriched matrix Γ^{e} by applying the following two rewrite rules with respect to Σ until no rules can be applied. A rule applies to a matrix Γ with respect to Σ iff the matrix matches the left hand side of the rule and the right hand side is a valid set description under the same variable assignment.*

   $(\phi_1, \phi_2 \in \mathcal{S}; \tau_1, \tau_2 \in \mathcal{T}; \alpha \in \mathcal{A}; \pi \in \mathcal{A}^*)$

$$\text{(1)}\quad \begin{array}{l} \Gamma' \uplus \left\{ \Sigma' \cup \left\{ \begin{array}{l} :\mathrm{IN}\pi \sim \phi_1, \\ :\mathrm{OUT}\pi \sim \phi_2 \end{array} \right\} \right\} \\ \uplus \left\{ \Sigma'' \cup \left\{ \begin{array}{l} :\mathrm{IN}\pi \sim \phi_1, \\ :\mathrm{OUT}\pi \sim \phi_1 \end{array} \right\} \right\} \end{array} \underset{\Sigma}{\overrightarrow{}} \Gamma' \cup \left\{ \Sigma'' \cup \left\{ \begin{array}{l} :\mathrm{IN}\pi \sim \phi_1, \\ :\mathrm{OUT}\pi \sim \phi_1 \end{array} \right\} \right\}$$

$$\mathit{if}\ \phi_1 \neq \phi_2 \quad \mathit{and}\ :\mathrm{OUT}\pi \sim \flat \notin \Sigma$$

$$\text{(2)}\quad \Gamma' \uplus \left\{ \Sigma' \uplus \left\{ \begin{array}{l} :\mathrm{OUT}\pi \sim \phi_1, \\ :\mathrm{IN}\pi \sim \phi_2 \end{array} \right\} \right\} \underset{\Sigma}{\overrightarrow{}} \Gamma' \cup$$

$$\left\{ \Sigma' \uplus \left\{ \begin{array}{l} :\mathrm{OUT}\pi \sim \phi_1, \\ :\mathrm{IN}\pi \sim \phi_2, \\ :\mathrm{IN}\pi\alpha \approx :\mathrm{OUT}\pi\alpha \end{array} \middle| \begin{array}{l} \mathit{approp}(\phi_1, \alpha) \cap \\ \mathit{approp}(\phi_2, \alpha) \neq \emptyset \end{array} \right\} \right\}$$

$$\mathit{if}\ :\mathrm{OUT}\pi\alpha \approx :\mathrm{OUT}\pi\alpha \notin \Sigma$$

3. *The frame enriched LRS matrix Γ^{lr} is the union of all frame enriched matrices Γ^{e} obtained, from which all inconsistent clauses and literals of the form $\tau \sim \sharp$ and $\tau \sim \flat$ $(\tau \in \mathcal{T})$ have been eliminated.*

Rule (1) is responsible for framing the species of paths the corresponding out-paths of which are mentioned in the out-specification. It checks if the type on a certain in-path is compatible with that on the corresponding out-path, i.e., it checks if the species of the in-path is assigned to both the in and the out-path in some disjunct. If that's the case, it eliminates the disjunct in which the in-path and the out-path are not assigned the same species. This rule relating the typing in the out-specification to the in-specifications is not applied for out-paths which are specified to be flattened, i.e., if $:\mathrm{OUT}\pi \sim \flat \in \Sigma$.

The second rule performs framing of all parts not mentioned in the out-specification. It introduces structure sharing between DLR-In and DLR-Out for all attributes α extending a path which is defined in LRS-Out in case α is appropriate for both the path in LRS-In and the corresponding one in DLR-Out and the path extended by α is not itself defined in DLR-Out. Note that the path extended by α will be defined in DLR-Out in case that path was specified with a flat or sharp instruction, thus keeping the rule from framing a path equality without requiring a special treatment for these instructions.

5.3. *An example*

To illustrate the formalization with a complex case of a lexical rule, let us take a look at an example taken from Pollard and Sag (1994), the Complement Extraction Lexical Rule (CELR). There are two reasons for looking at this example. On the one hand the signature is explicitly given by Pollard and Sag. This is necessary to understand what goes on with a lexical rule specification. On the other hand, the CELR is rather difficult to express without a formalized lexical rule mechanism and can cause unwanted results under some interpretations as discussed by Höhle (1995). So this makes it a good test case to see whether we've made things any clearer, even though a lot of the possibilities which we envisaged in the design of the lexical rule specification language will naturally play no role in this particular case.

The CELR as provided by Pollard and Sag (1994:378) which we already briefly mentioned in the discussion around Fig. 9 is repeated in Fig. 39 below.

This original specification is written down using a number of shorthands, such as abbreviated feature paths and the use of "...". To clarify the intended interpretation of the rule, Pollard and Sag (1994: fn. 36) write "The intended interpretation of the lexical rule is that all occur-

$$\begin{bmatrix} \textsc{arg-st} & \langle \ldots, \boxed{3}, \ldots \rangle \\ \textsc{comps} & \langle \ldots, \boxed{3}[\textsc{loc}\ \boxed{1}], \ldots \rangle \\ \textsc{inher}|\textsc{slash} & \boxed{2} \end{bmatrix} \mapsto \begin{bmatrix} \textsc{arg-st} & \langle \ldots, \boxed{4}[\textsc{loc}\ \boxed{1}, \textsc{inher}|\textsc{slash}\{\boxed{1}\}], \ldots \rangle \\ \textsc{comps} & \langle \ldots\ldots \rangle \\ \textsc{inher}|\textsc{slash} & \{\boxed{1}\} \cup \boxed{2} \end{bmatrix}$$

Fig. 39. The CELR as specified by Pollard and Sag (1994).

rences of $\boxed{3}$ in the input (except for the one on the COMPS list, which is eliminated in the output) are to be replaced in the output by a specification $\boxed{4}$, which is exactly like $\boxed{3}$ except that it bears the additional specification $[\textsc{inher}|\textsc{slash}\{\boxed{1}\}]$. This will ensure, for example, that in the case of a raising-to-object verb, the complement subject synsem object remains token-identical with the SUBCAT element corresponding to the "extracted" subject (e.g., in who I expected _ to come.)."

As a first step towards formalizing this lexical rule specification, we need to eliminate the informal shorthands. As explicit representation, we obtain the probably intended lexical rule specification shown in Fig. 40.

$$\begin{bmatrix} \textsc{arg-st}\ \boxed{7} \oplus \langle \boxed{3} \mid \boxed{8} \rangle \\ \textsc{synsem} \begin{bmatrix} \textsc{loc}|\textsc{cat}|\textsc{val}|\textsc{comps} & \boxed{6} \oplus \langle \boxed{3}[\textsc{loc}\ \boxed{1}] \mid \boxed{5} \rangle \\ \textsc{nloc}|\textsc{inher}|\textsc{slash} & \boxed{2} \end{bmatrix} \end{bmatrix} \mapsto \begin{bmatrix} \textsc{arg-st}\ \boxed{7} \oplus \left\langle \begin{bmatrix} \textsc{loc}\ \boxed{1} \\ \textsc{nloc}|\textsc{inher}|\textsc{slash}\{\boxed{1}\} \end{bmatrix} \mid \boxed{8} \right\rangle \\ \textsc{synsem} \begin{bmatrix} \textsc{loc}|\textsc{cat}|\textsc{val}|\textsc{comps} & \boxed{6} \oplus \boxed{5} \\ \textsc{nloc}|\textsc{inher}|\textsc{slash} & \{\boxed{1}\} \cup \boxed{2} \end{bmatrix} \end{bmatrix}$$

Fig. 40. An explicit version of the CELR.

In eliminating the "..." notation we, however, had to introduce the operator $\oplus$ for the append relation and we left the $\cup$ operator for the set union relation from the original specification. Since we based our lexical rule specification language on SRL, which does not provide such relations as first class citizens, we would need to introduce these relations into our ontology and refer to them using a so-called junk-slot encoding of relations (Aït-Kaci 1984, King 1992, Carpenter 1992). Alternatively, one could redefine the lexical rule specification language and its interpretation to be based on an extension of SRL with relations as provided by the Relational Speciate Re-entrant Language (RSRL) (Richter et al. 1999, Richter 1999). Since the different ways to encode relations in HPSG are a separate issue and we do not want to complicate the example further with a junk-slot encoding of the relations append and union, we base our illustration on a simplified version of the CELR. Instead of treating any element on COMPS and ARG-ST with any SLASH set, the particular instance of the CELR we discuss extracts the second element on COMPS and ARG-ST for entries with an empty SLASH set.

Fig. 41 shows how the simplified CELR can be specified by the linguist using our setup. As usual with lexical rules, only those parts which

$$\begin{bmatrix} \textsc{arg-st}|\textsc{rest}|\textsc{first}\ \boxed{3} \\ \textsc{synsem} \begin{bmatrix} \textsc{loc}|\textsc{cat}|\textsc{val}|\textsc{comps}|\textsc{rest}\ \langle \boxed{3}[\textsc{loc}\ \boxed{1}] \mid \boxed{5} \rangle \\ \textsc{nloc}|\textsc{inher}|\textsc{slash}\{\} \end{bmatrix} \end{bmatrix} \mapsto \begin{bmatrix} \textsc{arg-st}|\textsc{rest}|\textsc{first}|\textsc{nloc}|\textsc{inher}|\textsc{slash}\{\boxed{1}\} \\ \textsc{synsem} \begin{bmatrix} \textsc{loc}|\textsc{cat}|\textsc{val}|\textsc{comps}|\textsc{rest}\ \boxed{5} \\ \textsc{nloc}|\textsc{inher}|\textsc{slash}\{\boxed{1}\} \end{bmatrix} \end{bmatrix}$$

Fig. 41. Lexical rule specification of a simplified CELR.

are intended to be changed need to be mentioned. No type or path equality elimination is needed for this example.

Since no typing information is specified in LRS-Out and those attributes which have types as values that have subtypes (HEAD, NUCLEUS, RESTIND, DTRS, etc.) are not mentioned in LRS-Out, all the work to map the CELR into a description is done by the rewrite rule that adds path equalities between the in- and the out-description. The DLR resulting from enriching the CELR is shown in Fig. 42.[27] To distinguish the tags present in the lexical rule specification from the tags representing path equalities which were added by the enriching algorithm, the latter start with number 10 and are marked in grey. Also, the attributes that were part of the original specification are underlined. For each of these defined paths normalization introduced a specification $:\tau \approx :\tau$ to mark which paths are defined and species specifications $:\tau \sim \phi$. So, the species along the underlined paths in Fig. 42 are introduced by normalization. The path equalities represented by the grey tags are introduced by enriching: Along the defined paths the appropriate attributes are introduced and path equalities between paths in the in-specification and the out-specification are added for those attributes which directly extend a defined path but are not themselves defined, i.e., underlined.

Note that the element on the ARG-ST list of the output which corresponds to the one that is extracted from the COMPS list turns out to be identical to the element on the input's ARG-ST list except for the NLOC|INHER|SLASH specification. This is just what was intended but what in the absence of a formalized lexical rule mechanism was not formally expressed in the original formulation of the CELR as discussed by Pollard and Sag (1994:378, fn. 36).

Finally, let us turn to the problem with the CELR discussed in Höhle (1995). It consists of the apparent need to modify certain path equalities before 'copying them over' from the in-specification to the out-specification. More concretely, assume a CELR removing the second element of a COMPS list applies to an input which includes a path equality involving the fourth element of the COMPS list, i.e., the path REST|REST|REST|FIRST. Since the second element is eliminated by the lexical rule, the path equality in the output has to refer to the third element (REST|REST|FIRST) instead of the fourth in order to involve the same entity as in the input. Simply transferring path equalities from the input to the output therefore

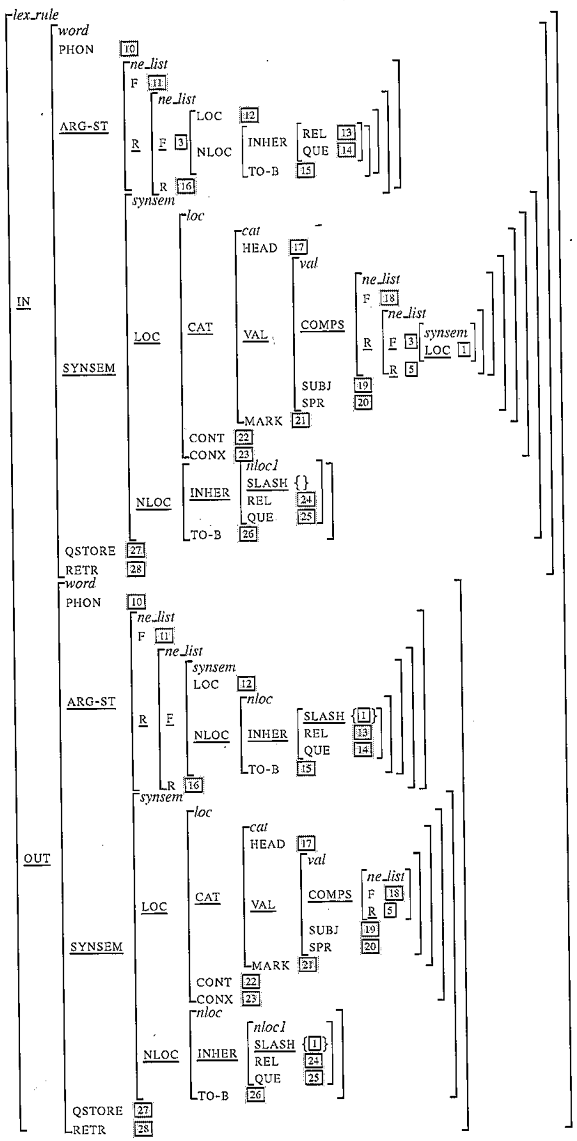


Fig. 42. The explicit DLR resulting from enriching the CELR of Fig. 41.

does not seem to provide the intended result. Looking over the characterization of the problem, it becomes apparent that it is closely tied to the traditional MLR interpretation of lexical rules as mapping between descriptions. Under the DLR formalization presented above, path equal-

ities between the in- and the out-specification ensure the framing of unchanged properties. This means that path equalities holding in the input are not transferred as such but as the result of the transitivity of path equality. Höhle's problem therefore does not arise under our DLR approach.

6. SUMMARY

In this paper, we discussed the status of the lexicon and the possibilities for expressing lexical generalizations in the paradigm of Head-Driven Phrase Structure Grammar. We showed that the architecture readily supports the use of lexical principles to express so-called vertical generalizations, i.e., generalizations over a class of word objects. We then turned to horizontal generalizations and investigated a possibility to formalize lexical rules based on SRL as a logical basis for HPSG. First, we defined lexical rules so that they can be constrained by ordinary descriptions. Then we explored and defined a lexical rule specification notation which allowed us to leave certain things implicit. Finally, we showed how we can get from the lexical rule specification to the explicit lexical rule constraints.

Even though the two approaches to interpreting an LRS we discussed, the MLR and the DLR approach, share many aspects, it is important to understand that the way in which these approaches do the actual interpretation is very different. In the MLR approach, an algorithm is supplied which, independent of the rest of the theory, takes a set of lexical entries, and constructs a (possibly infinite) set of derived lexical entries resulting from lexical rule application. In the DLR approach, the interpretation of an LRS is divided into two steps: First, the LRS is transformed into an ordinary constraint which is integrated into the theory. The real interpretation of the LRS as a relation extending the set of grammatical word objects is left to the second step, where the whole theory is interpreted in the ordinary way.

We believe there are some nice properties of such a DLR approach: First of all, apart from the mapping from the specification to explicit constraints, we did not add any additional machinery to the logic. The semantics of the lexical rule specification after the mapping is provided by the ordinary definition of the interpretation of an HPSG theory in SRL. The advantage this has for the linguist is that when it comes down to seeing exactly what a certain lexical rule specification means, (s)he can always take a look at the resulting enriched, fully explicit descriptions of lexical rules in the language used to write the rest of the HPSG theory, instead of having to interpret the lexical rule specification directly in some kind of additional formal system.

Second, the mapping from lexical rule specifications to explicit constraints is done independent of the lexical entries. It suffices to look at a lexical rule specification and the signature to determine what remained implicit in the lexical rule specification and how it can be made explicit. This is possible because HPSG is built on a typed feature logic and a closed word interpretation of a type hierarchy.

Third, the approach presented is highly modular and adaptable to the linguist's needs: One can decide on the data structure for lexical rules one likes best (relations or ordinary descriptions), alter/extend the lexical rule specification language in a way one likes, and alter/extend the rewrite rules which enrich lexical rule specifications to ordinary descriptions in a way one likes. This is important until a real discussion of possibilities and linguistic consequences of various setups has shown what linguists working in HPSG really want to write down and what it's supposed to mean.

And finally, taking descriptions of lexical rule objects as underlying encoding in the way proposed in this paper makes it possible to hierarchically group lexical rules and express constraints on (groups of) lexical rules. This allows us to express general principles every lexical rule has to obey, and it makes it possible to express that a group of lexical rules shares certain properties.

ACKNOWLEDGEMENTS

I thank Thilo Götz, Erhard Hinrichs, Tilman Höhle, Paul King, Guido Minnen, and Bill Rounds for valuable feedback, the two anonymous reviewers for their helpful comments, and particularly Mike Calcagno and Carl Pollard for interesting discussions and cooperation on the topic of lexical rules.

NOTES

1 On the computational side, the idea to express macros just like other kinds of relations is incorporated in the ConTroll system (Götz & Meurers 1995, 1997a,b). To be efficient this requires a dedicated computational treatment of deterministic relations.

2 That a meta-level is involved here is clearly expressed in the discussion of *descriptive adequacy*, where Chomsky (1965:24) states that "a linguistic theory must contain a definition of 'grammar,' that is, a specification of the class of potential grammars."

3 The mnemonic names often given to macros can give the impression of non-arbitrariness to such abbreviations. On formal grounds, however, a macro is nothing but an arbitrary symbol representing an arbitrary collection of descriptions. A discussion of the parallel misuse of such naming schemes in Artificial Intelligence can be found in McDermott (1981).

4 The signature consists of the type hierarchy specifying what type of objects we will talk about and the appropriateness conditions declaring which properties of which type of objects we want to include in our model. The signature thus defines the vocabulary that can be used when writing down the theory.

5 Where to introduce a new distinction in case the ontology does not yet make it possible to pick out the relevant class of objects depends on the particular kind of class which is to be singled out. For example, for lexical classes which are subclasses of categorial

distinctions, the most appropriate location would be to introduce them as subtypes in the hierarchy below *head*, where certain categorial distinctions are already encoded.

6 For space reasons, some of the attribute names are abbreviated in the figure.

7 Note that the mnemonic names given to such types formally have no more meaning than a simple constant like *t*. McDermott (1981) discusses the inherent danger of such naming schemes, which give rise to the fidelity fallacy: an observer believes that what a symbol actually denotes within a formal system is what the observer expects it to denote (Murray 1995:8).

8 Alternatively, one could turn the implications (→) in Fig. 2 into biconditionals (↔). This would create complex antecedents though, which defeats the original mission to use only type antecedents.

9 The principles in Fig. 3 make use of negation (¬) to compactly single out the complement of the consequents of Fig. 2. Under the standard closed-world assumption, these negations can be eliminated by disjunctively enumerating all possibilities.

10 Briscoe and Copestake (1999) provide an interesting discussion of lexical rules in a typed *default* feature structure framework (Lascarides et al. 1996, Lascarides & Copestake 1999), which is an extension of a *Kasper-Rounds* logic (Rounds & Kasper 1986, Moshier & Rounds 1987, Carpenter 1992). The ontological assumptions and formal properties of a Kasper-Rounds logic differ in crucial respects from those of an *Attribute-Value* logic (Johnson 1988, Smolka 1988, King 1989), and King (1994) shows that only the latter is compatible with the assumptions of HPSG as proposed in Pollard and Sag (1994). Since it is unclear how defaults could be integrated into an Attribute-Value logic and therefore into the setup of HPSG discussed here, a discussion of default formalizations of lexical rules is beyond the scope of this paper.

11 The question how to process with lexical rules as formalized in this paper is not discussed here; it is the topic of Meurers and Minnen (1997).

12 Please note the terminology used here and throughout the paper: The *lexicon* is a collection of lexical entries and each *lexical entry* is a description of a set of word objects. Sometimes we will simply speak of *words* when we mean word objects (but never for lexical entries).

13 We renamed the SUBCAT attribute of the original lexical rule to the now more common ARG-ST.

14 One could attempt to extend SRL to allow infinite disjunctions, where each of the disjuncts can be recursively enumerated, but such an extension is beyond the scope of this paper.

15 The valence requirement is *token-identical* to (part of) the realized argument, i.e., it points to the same object.

16 Note that nothing ensures that the *synsem* object on the valence attribute of a word is part of a grammatical sign at all.

17 As an alternative to the standard HPSG setup, Hinrichs and Nakazawa (1994) propose to represent and identify the entire *sign* value of each argument. In their setup, looking at a single word at the leaf of a tree therefore reveals all information about the word and all its arguments. To a large degree this eliminates the distinction between a lexical and a syntactic level of explanation.

18 Note that nothing we have stated here changes the distinction between syntactic transformations and lexical rules discussed above. In a syntactic transformation a word has to exist as part of a grammatical syntactic tree, whereas this is not the case for lexical rules. This is still true here, since we only require that the word that is the input to the lexical rule is grammatical and not that it constructs in a grammatical syntactic tree.

19 There is an exception to this statement which arises when one specifies a meta-level lexical rule which includes the complete input as part of the output, e.g., by introducing an extra attribute for words which in the output of the lexical rule is specified to be identical to the complete input. However, since one of the motivations for a meta-level formalization of lexical rules is to avoid representing the source of the derivation as part of the model, this possibility is not very attractive and would only amount to specifying DLRs in an indirect way.

20 Whether or not such forms should be derived by a lexical rule at all is an independent issue which we skip here for the sake of the argument.
21 In this and the following figures only the X type values are shown.
22 Following a reviewer's suggestion, we do not generally introduce meta-variables into the description language. While it would allow us to keep the specification language uniform for both a DLR and an MLR approach, the different interpretation of the two approaches in practice already makes it necessary to chose between them.
23 One could avoid splitting up the LRS by adding type equality as syntactic sugar to SRL. But as this is only one of several problematic aspects discussed here, we will not pursue this possibility.
24 For easier comparison with standard HPSG notation, one can introduce a finite join semi-lattice $\langle \mathcal{Z}, \preceq \rangle$ as type hierarchy with $\mathcal{Z} \supset \mathcal{S}$. A type assignment is then simply an abbreviation for a set of variety assignments: $\tau \sim t = \bigvee\{\tau \sim \phi | \phi \preceq t\}$ with $\tau \in \mathcal{T}, t \in \mathcal{Z}$, and $\phi \in \mathcal{S}$. In this paper, we assume every type assignment to be expanded in that way. Nothing of theoretical importance hinges on this.
25 We write $\{X \rightharpoonup Y\}$ for the set of partial functions from set X to set Y.
26 Since the path equality relation is transitive, there are several possibilities to encode the example in the formal notation. Normalization (cf., section 5.2.1) introduces all path equalities which can be inferred due to transitivity.
27 To take a place in the theory, the description in Fig. 42 is included as one of the disjuncts on the right-hand side of the constraint on *lex_rule* which we saw in Fig. 14. In the figure, the feature names F and R are used as abbreviated notation for the *ne_list* attributes FIRST and REST.

REFERENCES

Aït-Kaci, H. 1984. A Lattice Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Bouma, G., Malouf, R. & Sag, I. A. 2001. Satisfying Constraints on Extraction and Adjunction. *Natural Language and Linguistic Theory* 19(1), 1–65. ftp://csli-ftp.stanford.edu/linguistics/sag/bms-nllt.ps.

Bresnan, J. 1982. Passive in Lexical Theory. In Bresnan, J. (ed.), *The Mental Representation of Grammatical Relations*. Cambridge, MA: MIT Press, pp. 3–86.

Briscoe, T. & Copestake, A. 1999. Lexical Rules in Constraint-based Grammars. *Computational Linguistics* 25(4), 487–526. http://www-csli.stanford.edu/~aac/papers/lr2-0.ps.gz.

Calcagno, M. 1995. Interpreting Lexical Rules. In *Proceedings of the First Conference on Formal Grammar*. Barcelona. http://ling.osu.edu/~dm/papers/calcagno-fg95.html.

Calcagno, M. & Pollard, C. 1995. *Lexical Rules in HPSG: What are they?* Ms., dated 17 July 1995, Linguistics Department, Ohio State University, Columbus, OH. http://ling.osu.edu/~dm/papers/calcagno-pollard.html.

Carpenter, B. 1991. The Generative Power of Categorial Grammars and Head-Driven Phrase Structure Grammars with Lexical Rules. *Computational Linguistics* 17(3), 301–314.

Carpenter, B. 1992. *The Logic of Typed Feature Structures – With Applications to Unification Grammars, Logic Programs and Constraint Resolution*, vol. 32 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge, UK: Cambridge University Press.

Chomsky, N. 1965. *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press.

De Kuthy, K. & Meurers, W. D., 2001. On Partial Constituent Fronting in German. *Journal of Comparative Germanic Linguistics* 3(3), 143–205. http://www.ling.osu.edu/~dm/papers/dekuthy-meurers-jcgl01.html.

Dörre, J. 1994. *Feature-Logik und Semiunifikation*. No. 48 in Arbeitspapiere des SFB 340. Stuttgart: Universität Stuttgart.

Flickinger, D. 1987. Lexical Rules in the Hierarchical Lexicon. Ph.D. thesis, Stanford University, Stanford, CA.

Gazdar, G., Klein, E., Pullum, G. K. & Sag, I. A. 1985. *Generalized Phrase Structure Grammar*. Cambridge, MA: Harvard University Press.

Geach, P. T. 1970. A Program for Syntax. *Synthese* 22, 483–497.

Gerdemann, D. 1995. Open and Closed World Types in NLP Systems. In *Proceedings of the DGfS Fachtagung Computerlinguistik*. Düsseldorf. http://www.sfs.uni-tuebingen.de/~dg/open_world.ps.

Gerdemann, D. & King, P. 1994. The Correct and Efficient Implementation of Appropriateness Specifications for Typed Feature Structures. In *Proceedings of the 15th Conference on Computational Linguistics (COLING-94)*. Kyoto, pp. 956–960. http://www.sfs.uni-tuebingen.de/~dg/correct.ps.

Ginsberg, M. L. (ed.). 1987. *Readings in Nonmonotonic Reasoning*. San Francisco, CA: Morgan Kaufmann Publishers.

Götz, T. 1994. *A Normal Form for Typed Feature Structures*. No. 40 in Arbeitspapiere des SFB 340, Tübingen: Universität Tübingen. Published version of Master's thesis, Seminar für Sprachwissenschaft, Universiät Tübingen. http://www.sfs.uni-tuebingen.de/~tg/thesis.ps.gz.

Götz, T. 2000. Feature Constraint Grammars. Ph.D. thesis, Seminar für Sprachwissenschaft, Universität Tübingen, Tübingen. http://w210.ub.uni-tuebingen.de/dbt/volltexte/2000/133.

Götz, T. & Meurers, W. D. 1995. Compiling HPSG Type Constraints into Definite Clause Programs. In *Proceedings of the 33nd Annual Meeting of the Association for Computational Linguistics (ACL 95)*. Cambridge, MA: MIT, pp. 85–91. http://ling.osu.edu/~dm/papers/acl95.html.

Götz, T. & Meurers, W. D. 1997a. The ConTroll System as Large Grammar Development Platform. In *Proceedings of the Workshop "Computational Environments for Grammar Development and Linguistic Engineering (ENVGRAM)" held in conjunction with the 35th Annual Meeting of the ACL and 8th Conference of the EACL*. Madrid: Universidad Nacional de Educación a Distancia, pp. 38–45. http://ling.osu.edu/~dm/papers/envgram.html.

Götz, T. & Meurers, W. D. 1997b. Interleaving Universal Principles and Relational Constraints over Typed Feature Logic. In *Proceedings of the 35th Annual Meeting of the ACL and 8th Conference of the EACL*. Madrid: Universidad Nacional de Educación a Distancia, pp. 1–8. http://ling.osu.edu/~dm/papers/acl97.html.

Hinrichs, E. W., Meurers, W. D. & Nakazawa, T. (eds.), 1994. *Partial-VP and Split-NP Topicalization in German – An HPSG Analysis and its Implementation*. No. 58 in Arbeitspapiere des SFB 340. Tübingen: Universität Tübingen.

Hinrichs, E. W. & Nakazawa, T. 1989. Flipped Out: Aux in German. In *Papers from the 25th Regional Meeting of the Chicago Linguistic Society*. Chicago, IL, pp. 193–202.

Hinrichs, E. W. & Nakazawa, T. 1994. Partial-VP and Split-NP Topicalization in German – An HPSG Analysis. In Hinrichs et al. (1994), pp. 1–46.

Höhfeld, M. & Smolka, G. 1988. *Definite Relations over Constraint Languages*. LILOG technical report 53, IBM Deutschland. http://www.ps.uni-sb.de/Papers/abstracts/LR-53.html.

Höhle, T. N. 1978. *Lexikalistische Sxntax. Die Aktiv-Passiv-Relation und andere Infinitkonstruktionen im Deutschen*. No. 67 in Linguistische Arbeiten. Tübingen: Max Niemeyer Verlag.

Höhle, T. N. 1983. *Topologische Felder*. Ms., Universität Köln, Köln.

Höhle, T. N. 1995. *The Complement Extraction Lexical Rule and Variable Argument Raising*. Handout for a talk given at the Int. HPSG Workshop 95, 21–23. June 1995, Universität Tübingen, Tübingen.

Höhle, T. N. 1996a. *Einf. HPSG: Die Grammatik, Generalisierungen übers Lexikon*. Handout dated 18/19. April 1996, Deutsches Seminar, Universität Tübingen, Tübingen.

Höhle, T. N. 1996b. *Remarks on the Word Principle*. Handout dated 25 June 1996, Deutsches Seminar, Universität Tübingen, Tübingen.

Johnson, M. 1988. *Attribute-value Logic and the Theory of Grammar*, vol. 16 of *CSLI Lecture Notes*. Stanford, CA: CSLI Publications.

Kepser, S. 1994. *A Satisfiability Algorithm for a Logic for Typed Feature Structures.* No. 60 in Arbeitspapiere des SFB 340. Tübingen: Universität Tübingen. Published version of Master's thesis, Seminar für Sprachwissenschaft, Universität Tübingen. http://www.sfs.uni-tuebingen.de/sfb/reports/berichte/60/60abs.html.

King, P. J. 1989. A Logical Formalism for Head-Driven Phrase Structure Grammar. Ph.D. thesis, University of Manchester, Manchester.

King. P. J. 1992. Unification Grammars and Descriptive Formalisms. Lecture notes, Seminar für Sprachwissenschaft, Universität Tübingen.

King, P. J. 1994. *An Expanded Logical Formalism for Head-driven Phrase Structure Grammar.* No. 59 in Arbeitspapiere des SFB 340. Tübingen: Universität Tübingen. http://www.sfs.uni-tuebingen.de/sfb/reports/berichte/59/59abs.html.

Lascarides, A., Briscoe, T., Asher, N. & Copestake, A. 1996. Order Independent and Persistent Typed Default Unification. *Linguistics and Philosophy* 19(1), 1–89. http://www-csli.stanford.edu/~aac/papers/landp.ps.gz and http://www.cogsci.ed.ac.uk/~alex/papers/pdu.ps.

Lascarides, A. & Copestake, A. 1999. Default Representation in Constraint-Based Frameworks. *Computational Linguistics* 25(1). 55–106. http://www-csli.stanford.edu/~aac/papers/yadu5-0.pdf and http://www.cogsci.ed.ac.uk/~alex/papers/yadu.ps.

McCarthy, J. & Hayes, P. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In Meltzer, B. & Michie, D. (eds.), *Machine Intelligence 4.* Edinburgh: Edinburgh University Press. Reprinted in Ginsberg (1987).

McDermott, D. 1981. Artificial Intelligence Meets Natural Stupidity. In Haugeland, J. (ed.), *Mind Design.* Cambridge, MA: MIT Press, pp. 143–160. First appeared 1976 in SIGART Newsletter 57.

Meurers, W. D. 1994. On Implementing an HPSG Theory – Aspects of the Logical Architecture, the Formalization, and the Implementation of Head-Driven Phrase Structure Grammars. In Hinrichs et al. (1994), pp. 47–155. http://ling.osu.edu/~dm/on-implementing.html.

Meurers, W. D. 1995. Towards a Semantics for Lexical Rules as used in HPSG. In *Proceedings of the ACQUILEX II Workshop on the Formalisation and Use of Lexical Rules.* Cambridge, UK, pp. 1–20. Also presented at the First Conference on Formal Grammar, Barcelona, 1995. http://ling.osu.edu/~dm/papers/dlrs.html.

Meurers, W. D. 1997. Using Lexical Principles in HPSG to Generalize over Valence Properties. In Kruijff, G.-J. M., Morrill, G. V. & Oehrle, R. T. (eds.), *Proceedings of the Third Conference on Formal Grammar.* Aix-en-Provence: Université de Provence, pp. 137–146. http://ling.osu.edu/~dm/papers/using-lexical-principles.html.

Meurers, W. D. 2000. *Lexical Generalizations in the Syntax of German Non-Finite Constructions.* No. 145 in Arbeitspapiere des SFB 340. Tübingen: Universität Tübingen. Ph.D. thesis, Universiät Tübingen. http://ling.osu.edu/~dm/papers/diss.html and http://w210.ub.uni-tuebingen.de/dbt/volltexte/2000/118.

Meurers, W. D. & Minnen, G. 1997. A Computational Treatment of Lexical Rules in HPSG as Covariation in Lexical Entries. *Computational Linguistics* 23(4), 543–568. http://ling.osu.edu/~dm/papers/meurers-minnen-97.html.

Miller, P. H. & Sag, I. A. 1993. French Clitic Climbing Without Clitics or Climbing. Ms., University of Lille and Stanford University. A substantially revised version appeared as Miller and Sag (1997).

Miller, P. H. & Sag, I. A. 1997. French Clitic Movement Without Clitics or Movement. *Natural Language and Linguistic Theory* 15(3), 573–639. ftp://csli-ftp.stanford.edu/linguistics/sag/french-clitic.ps.gz.

Monachesi, P. 1999. *A Lexical Approach to Italian Cliticization.* No. 84 in Lecture Notes series. Stanford, CA: CSLI Publications. Published version of 1995 Ph.D. thesis, University of Utrecht, Utrecht.

Moshier, M. A. & Rounds, W. C. 1987. A Logic for Partially Specified Data Structures. In *Proceedings of the 14th ACM Symposium on Principles of Programming Languages. München.* pp. 156–167.

Murray, K. S. 1995. Learning as Knowledge Integration. Ph.D. thesis, University of Texas, Austin, TX. Published as CS-TR-95-41. ftp://ftp.cs.utexas.edu/pub/techreports/tr95-41.ps.Z.

Pollard, C. & Sag, I. A. 1987. *Information-based Syntax and Semantics, Vol. 1: Fundamentals.* No. 13 in CSLI Lecture Notes. Stanford, CA: CSLI Publications.

Pollard, C. & Sag, I. A. 1994. *Head-Driven Phrase Structure Grammar.* Chicago, IL: University of Chicago Press.

Richter, F. 1997. Die Satzstruktur des Deutschen und die Behandlung langer Abhängigkeiten in einer Linearisierungsgrammatik. Formale Grundlagen und Implementierung in einem HPSG-Fragment. In Hinrichs, E. W., Meurers, W. D., Richter, F., Sailer, M. & Winhart, H. (eds.), *Ein HPSG-Fragment des Deutschen. Teil 1: Theorie,* Tübingen: Universität Tübingen, no. 95 in Arbeitspapiere des SFB 340, pp. 13–187. http://ling.osu.edu/~dm/papers/sfb-report-nr-95/kapitel2-richter.html.

Richter, F. 1999. RSRL for HPSG. In Kordoni, V. (ed.), *Tübingen Studies in Head-Driven Phrase Structure Grammar.* Tübingen: Universität Tübingen, Arbeitspapiere des SFB 340 Nr. 132, pp. 74–115. http://www.sfs.uni-tuebingen.de/~fr/cards/rsrl4hpsg.html.

Richter, F. 2000. A Mathematical Formalism for Linguistic Theories with an Application in Head-Driven Phrase Structure Grammar. Ph.D. thesis, Seminar für Sprachwissenschaft, Universität Tübingen, Tübingen. http://www.sfs.uni-tuebingen.de/~fr/cards/dissertation.html.

Richter. F., Sailer, M. & Penn, G. 1999. A Formal Interpretation of Relations and Quantification in HPSG. In Bouma, G., Hinrichs, E. W., Kruijff, G.-J. M. & Oehrle, R. T. (eds.), *Constraints and Resources in Natural Language Syntax and Semantics.* Stanford, CA: CSLI Publications.

Riehemann, S. 1993. Word Formation in Lexical Type Hierarchies: A Case Study of *bar*-Adjectives in German. Master's thesis, Universität Tübingen, Tübingen. Published as SfS-Report 02-93. http://doors.stanford.edu/~sr/sfsreport.ps.

Rounds, W. C. & Kasper, R. T. 1986. A Complete Logical Calculus for Record Structures Representing Linguistic Information. In *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science, Cambridge, MA, USA.* pp. 38–43.

Sag, I. A. 1997. English Relative Clause Constructions. *Journal of Linguistics* 33(2), 431–484. ftp://csli-ftp.stanford.edu/linguistics/sag/rel-pap.ps.gz.

Smolka, G. 1988. *A feature logic with subsorts.* LILOG-Report 33, IBM Deutschland, Stuttgart. http://www.ps.uni-sb.de/Papers/abstracts/LR-33.html.

Van Noord, G. & Bouma, G. 1994. The Scope of Adjuncts and the Processing of Lexical Rules. In *Proceedings of the 15th Conference on Computational Linguistics (COLING-94).* Kyoto, pp. 250–256. http://xxx.lanl.gov/abs/cmp-lg/9404011.