Constraint-Based RMRS Construction from Shallow Grammars

Anette Frank, Kathrin Spreyer,
Witold Drożdżyński, Hans-Ulrich Krieger and Ulrich Schäfer

Language Technology Lab
German Research Center for Artificial Intelligence
DFKI GmbH
Saarbrücken, Germany
{frank,kspreyer,witold,krieger,uschaefer}@dfki.de

Proceedings of the HPSG04 Conference

Center for Computational Linguistics
Katholieke Universiteit Leuven

Stefan Müller (Editor)

2004

CSLI Publications

http://csli-publications.stanford.edu/

Constraint-Based RMRS Construction from Shallow Grammars

Abstract

We present a constraint-based syntax-semantics interface for the construction of RMRS (Robust Minimal Recursion Semantics) representations from shallow grammars. The architecture is designed to allow modular interfaces to existing shallow grammars of various depth—ranging from chunk grammars to context-free stochastic grammars. We define modular semantics construction principles in a typed feature structure formalism that allow flexible adaptation to alternative grammars and different languages.

1 Introduction

Semantic formalisms such as UDRT (Reyle, 1993), CLLS (Egg et al., 2001), or MRS (Copestake et al., 2003) provide elegant solutions for the treatment of semantic ambiguities in terms of underspecification—most prominently scope. In recent work, Copestake (2003) has investigated a novel aspect of underspecification in the design of semantic formalisms, which is concerned with the representation of *partial* semantic information, as it might be obtained from shallow, i.e., incomplete syntactic analysis. The main rationale for this type of underspecification is to ensure monotonicity, and thus upwards compatibility of the output of shallow parsing with semantic representations obtained from full syntactic parsing. Thus, Copestake's design of RMRS—Robust Minimal Recursion Semantics—provides an important contribution to a novel line of research towards integration of shallow and deep NLP. While previous accounts (Daum et al., 2003; Frank et al., 2003a) focus on shallow-deep integration at the syntactic level, Copestake aims at integration of shallow and deep NLP at the level of semantics.

In this paper we review the RMRS formalism designed by Copestake (2003) and present an architecture for a principle-based syntax-semantics interface for RMRS construction from shallow grammars. We argue for a unification-based approach to RMRS construction, to account for (underspecified) argument binding in languages with morphological as opposed to structural argument identification. We propose a reparsing architecture for RMRS construction that is especially designed to support flexible adaptation to different types of shallow to intermediate-level syntactic grammars that may serve as a basis for RMRS construction. We define modular semantics construction principles in a typed feature structure (TFS) formalism (Carpenter, 1992), which favours the portability to new grammars and languages. A challenge for principle-based semantics construction from shallow

[†] The research reported here was conducted in the project QUETAL, funded by the German Ministry for Education and Research, BMBF, under grant no. 01 IW C02. Thanks go to Ann Copestake and Dan Flickinger for discussion of our work, and to the audience of the HPSG Workshop on Semantics in Grammar Engineering, for interesting comments and questions.

grammars is the flat and sometimes non-compositional nature of the structures they typically produce. We propose RMRS semantics construction principles that can be applied to flat syntactic structures with various degrees of partiality.

The paper is structured as follows. Section 2 introduces the RMRS formalism. Section 3 gives an overview of the architecture we propose for RMRS construction from shallow grammars. We argue for a modular, constraint-based semantics construction module in a reparsing architecture, which we realise in the unification-based finite-state processing platform SProUT (Becker et al., 2002; Drozdzynski et al., 2004). In Section 4, we present the principles we define for morphological disambiguation and semantics construction from shallow grammars. Section 5 concludes and compares our work to alternative approaches.

2 RMRS—A Formalism for Partial Semantic Representation

Copestake (2003) presents a formalism for partial semantic representation that is derived from Minimal Recursion Semantics (MRS) (Copestake et al., 2003). Robust Minimal Recursion Semantics is designed to support novel forms of integrated shallow and deep NLP, by accommodating semantic representations produced by NLP components of various degrees of partiality and depth of analysis—ranging from PoS taggers and NE recognisers over chunk and (non-)lexicalised context-free grammars to deep grammars like HPSG with MRS output structures.

The advantages of a variable-depth semantic analysis are most evident for applications with conflicting requirements of robustness and accuracy. Given a range of NLP components of different depths of analysis that deliver compatible semantic representations, we can apply flexible integration methods: apply voting techniques, or combine partial results from shallow and deep systems (Copestake, 2003).

To allow intersection and monotonic enrichment of the output representations from shallow systems on one extreme of the scale with complete representations of deep analysis on the other, the missing specifications of the weakest system must be factored out from the most comprehensive deep representations. In the RMRS formalism, this concerns the following main aspects of semantic information:

Argument encoding. A ‘Parsons-style‘ notation accommodates for partiality of shallow systems wrt. argument identification. Instead of predicates with fixed arity, e.g., $l4{:}on(e',e,y)$, predicates and arguments are represented as independent elementary predications: $on(l4,e')$, $ARG1(l4,e)$, $ARG2(l4,y)$. This accounts for the uncertainty of argument identification in shallow grammars. Underspecification with respect to the type of argument is modeled in terms of a hierarchy over disjunctive argument types: $ARG1 \sqsubset ARG12$, $ARG2 \sqsubset ARG12$, $ARG12 \sqsubset \ldots \sqsubset ARGn$.

Variable naming and equalities. Constraints for equality of variables in elementary predications are to be added incrementally, to accommodate for knowledge-poor systems like PoS taggers, where the identity of referential variables of, e.g., adjectives and nouns in potential NPs cannot be established, or else chunkers, where the binding of arguments to predicates is only partially established.

An example. The following example of corresponding MRS (1.a) and RMRS (1.b) representations illustrates these differences (cf. Copestake, 2003).

(1) *Every fat cat sat on a mat*

a. MRS representation:
l0:every(x,h1,h2), l1:fat(x), l2:cat1(x), l3:CONJ, l4:sit1(e_{spast},x), l14:on2(e′,e,y), l9:CONJ, l5:some(y,h6,h7), l6:table1(y), qeq(h1,l3), qeq(h6,l6), in-g(l3,l1), in-g(l3,l2), in-g(l9,l4), in-g(l9,l14)

b. RMRS representation:
l0:every(x0), RSTR(l0,h1), BODY(l0,h2), l1:fat(x1), l2:cat1(x2), l3:CONJ, l4:sit1($e3_{spast}$), ARG1(l4,x2), l14:on2(e4), ARG1(l14,e3), ARG2(l14,x5), l9:CONJ, l5:some(x5), RSTR(l5,h6), BODY(l5,h7), l6:table1(x6), qeq(h1,l1), qeq(h6,l6), in-g(l3,l1), in-g(l3,l2), in-g(l9,l4), in-g(l9,l14), x0 = x1, x1 = x2, x5 = x6

3 An Architecture for RMRS Construction from Shallow Grammars

We aim at a modular syntax-semantics interface for RMRS construction that can be adapted to a wide range of *existing* shallow grammars, such as off-the-shelf chunk parsers or probabilistic (non-)lexicalised PCFGs. Moreover, we aim at the construction of underspecified, but *maximally constrained (i.e., resolved)* RMRS representations from shallow grammars.

A unification-based account. Chunk parsers and PCFG parsers for sentential structure do in general not provide functional information that can be used for argument identification. While in languages like English argument identification is to a large extent structurally determined, in other languages arguments are (partially) identified by case marking. In case-marking languages, morphological agreement constraints can yield a high degree of completely disambiguated constituents, as shown by Hinrichs and Trushkina (2002) for German. That is, by morphological disambiguation we can obtain maximally constrained identification of arguments from shallow analyses (see also Müller, 2004). We therefore propose a *unification-based approach* for RMRS construction, where agreement constraints can perform morphological disambiguation, and thus partial (i.e., underspecified) argument identification in case-marking languages.

In addition, by interfacing shallow analysis with morphological processing, we can infer important semantic features for referential and event variables, such as PNG and TENSE information. Thus, morphological processing can also be beneficial for languages with structural argument identification.

A reparsing architecture. In order to realise a *modular* interface to existing parsers, we follow a reparsing approach: For semantics construction, we extract constituency information from the output structure of a shallow parser, and deterministically reparse the original input string, while applying RMRS construction principles to the recomposed syntactic structures.

The advantages of a reparsing architecture—as opposed to a grammar with integrated syntactic and semantic rules—are that modular semantics construction rules can be adapted to the output structures of alternative existing parsers, including statistical parsers. Similarly, modular semantics construction rules can be ported to other languages, and applied to the output structures of existing chunkers or parsers for such languages.

Constraint-based RMRS construction—using cascaded SProUT. We define constraint-based principles for RMRS construction in a typed feature structure formalism. These semantics construction principles are applied to the (reparsed) syntactic structures provided by shallow parsing. In the reparsing step the constraints are resolved, to yield maximally specified RMRS representations.

The RMRS construction principles are defined and processed in the SProUT processing platform (Becker et al., 2002; Krieger et al., 2004). The SProUT system combines finite-state technology with unification-based processing. It allows the definition of finite-state transduction rules that apply to (sequences of) typed feature structures (TFSs), as opposed to atomic symbols. The left-hand side of a transduction rule specifies a regular expression over TFSs as a (longest-match) recognition pattern; the right-hand side specifies the output in terms of a typed feature structure. Regular expression operators are ? (optionality), $*$, + (Kleene star and plus), and $\{n,m\}$ (constrained iteration). Figure 1 displays a SProUT rule for the recognition of an NP consisting of an optional determiner, any number of adjectives and a noun. Coreferences (#) enforce unification of the referenced feature values. In the example, this enforces agreement of determiner, adjective and noun.

```
np :> morph & [POS art, INFL [CASE #case, NUM #num, GEND #gend]]?
      morph & [POS adj, INFL [CASE #case, NUM #num, GEND #gend]]*
      morph & [POS noun, INFL [CASE #case, NUM #num, GEND #gend]]
   -> phrase & [CAT np, AGR [CASE #case, NUM #num, GEND #gend]].
```

Figure 1: Example of a SProUT rule (cf. Krieger et al., 2004).

The rewrite rules are interfaced with a hierarchy of typed feature structures. In Figure 1, the rule is constrained to apply to feature structures of type *morph*; the output structure is defined to be of type *phrase*. The corresponding hierarchy of typed feature structures is specified separately from the rules.

The SProUT system offers a number of special features that proved extremely useful for our purposes.

Most importantly, the system has been extended to cascaded processing, such that the output of a set of rule applications (viz., TFSs) can provide the input to another set of rewrite rules, again on TFSs. This allows us to realise a cascade of grammars for lexical and phrasal RMRS construction, which we describe in more detail in Section 4.

Since SProUT operates on typed feature structures, we can define a hierarchy of types that facilitates the concise definition of semantics construction rules.

In SProUT, several distinct rules can simultaneously apply to the same sequence of input items, as long as the same (maximal) sequence of structures is matched. The output structures defined by the individual rules can then be unified, by special interpreter settings. This allows us to state modular RMRS construction principles with general application conditions that interact to yield complete RMRS structures.

The system offers a mechanism for rule prioritisation that implements *defaults*: rules can be (strictly)[1] ordered according to their priority, such that a rule with lower priority can only apply in case no rule with higher priority could be applied to the same input structure.

Finally, SProUT permits the definition of so-called *functional operators* to impose additional constraints for the application of a rule. Functional operators may extend the formal power of typed unification, and will be used for the implementation of constraining equations in argument identification rules (cf. Section 4.3).

Cascaded reparsing with SProUT. For cascaded reparsing, SProUT first performs morphological lookup on the original input string, which yields as output a list of TFSs of type *morph*. The morphological information is organised in a type hierarchy with disjunctive subtypes to underspecify ambiguities of inflectional features, e.g., case (see Krieger and Xu, 2003, and below).

The output sequence of morphological TFSs is input to the next cascade levels that perform morphological disambiguation and phrase composition.

For cascaded reparsing, or phrase composition according to the output structure of a shallow (context-free) parser, we enrich the input TFSs with constituency information that we extract from the parse tree for the corresponding input span: for each node we extract a uniquely referring node identifier (ID), together with the identifier (M-ID) and category (M-CAT) of its mother node. This implicitly encodes the necessary information about phrasal constituency that can be used to guide phrase composition and concurrent semantics construction in reparsing with SProUT. As unique node identifiers, we use word/phrase span information, as indicated in Figure 2.[2]

[1] Extensions for specification of *partial* ordering of rules are under way.

[2] Alternatively, one could use character position spans for node identification.

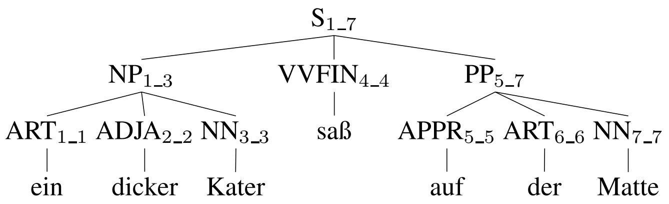

Figure 2: Indexed syntactic tree: *Ein dicker Kater saß auf der Matte* – A fat cat sat on the mat.

```
phrase :> synsem & [NODE [M-ID #mid, M-CAT #mcat]]+
         –> phrase  & [NODE [ID #mid], [M-SYN [CAT #mcat]].
```

Figure 3: Reparsing rule.

A general reparsing rule, displayed in Figure 3, is applied to the enriched input sequence of TFSs for lexical or phrasal nodes and produces as output a TFS for the implicitly defined mother node. The rule specifies that for all nodes in the matched input sequence,[3] their mother node identifier and category features (M-ID, M-CAT) must be identical, and defines the output (mother) node's local identifier and category feature (ID, CAT) by use of co-references (#mid, #mcat). Since the system obeys a longest-match strategy, the regular expression is constrained to apply to the same constituents as in the original parse tree.

Cascaded reparsing first applies to the sequences of leaf nodes that are provided by morphological processing. The output node sequence is enriched with the phrase-building information from the original parse tree, and is input to the phrase building and semantics construction rules. For phrase composition we define a cyclic cascade, where the output of a cascade is fed in as input to the same rules. The cycle terminates when no more phrase building rules could be applied to the input, i.e., the root category has been derived. This establishes a kind of fixpoint construction.

Morpho-syntactic disambiguation. In reparsing, we define very general principles for morpho-syntactic agreement, by defining agreement between single daughter constituents and their mother node, for categories like determiner, adjective, or noun (see Figure 4). This is in contrast to the usual definition of agreement rules between siblings. Since in our reparsing approach constituency is already predefined, the agreement constraints can be stated independently from precedence patterns for the recognition of different types of NPs. Defining morphological agreement independently for possibly occurring daughter constituents yields few and very general (disjunctive) projection principles that can also apply to "unseen" constituent sequences.

The rule in Figure 4 again exploits the longest-match strategy to constrain ap-

[3] with *synsem* a supertype of *lex* and *phrase*, see Section 4.1.

```
agr :>  lex & [NODE [M-ID #mid]]*
        ( lex & [NODE [M-ID #mid], M-SYN [CAT nn, AGR #agr]] |
          lex & [NODE [M-ID #mid], M-SYN [CAT adja, AGR #agr]] |
          lex & [NODE [M-ID #mid], M-SYN [CAT art, AGR #agr]] )
          lex & [NODE [M-ID #mid]]*
    ->  phrase & [NODE [ID #mid], M-SYN [AGR #agr]].
```

Figure 4: Modular (disjunctive) agreement projection rules.

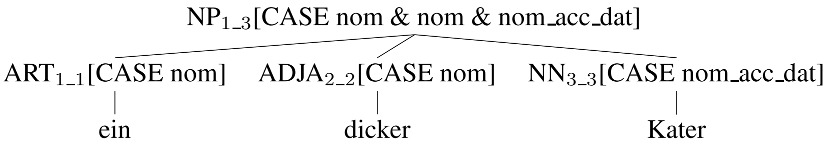
NP$_{1\_3}$[CASE nom & nom & nom_acc_dat]
ART$_{1\_1}$[CASE nom]
ADJA$_{2\_2}$[CASE nom]
NN$_{3\_3}$[CASE nom_acc_dat]
ein
dicker
Kater

Figure 5: Interaction of morphological constraints.

plication to the pre-defined constituents, by specifying coreferent M-ID features for all nodes in the rule's input sequence. In reparsing, the (possibly disjunctive) morphological types in the output structure of the individual rule applications are unified, yielding partially resolved inflectional features for the mother node. For NP$_{1\_3}$ in Figure 2, e.g., we obtain CASE *nom* by unification of *nom* (from ART$_{1\_1}$ and ADJA$_{2\_2}$) and *nom_acc_dat* (from NN$_{3\_3}$), see Figure 5. This resolved case value of the NP can be used for (underspecified) argument binding in RMRS construction (as discussed in more detail in Section 4.3).

Architecture of the SProUT-XSLT RMRS cascade. SProUT cascades can be defined using the declarative system description language SDL (Krieger, 2003). The sequence of SProUT cascade stages described in this paper has been specified in SDL and integrated into the 'Heart of Gold' (HoG) NLP architecture of Callmeier et al. (2004). HoG provides an XML-based architecture framework for the integration of deep and shallow NLP components. The declaratively defined SDL description of the cascade is compiled into a Java class which is integrated in a HoG architecture instance as a sub-architecture module (Figure 7).

The cascade, displayed in Figure 6, consists of four SProUT grammar instances with four interleaved XSLT transformations. The recursive application of phrase composition rules is defined by means of a cyclic SDL star operator. XSLT is used, e.g., to merge SProUT-generated structures with XML-encoded analyses of the chunk parser Chunkie (Skut and Brants, 1998). Motivation for and further details on XSLT transformation of typed feature structure representations are presented in Schäfer (2004).

```
chunkiermrs = ( sprout_rmrs_morph + xslt_pos_filter + sprout_rmrs_lex
                + ( xslt_nodeid_cat + sprout_rmrs_phrase )*
                + sprout_rmrs_final + xslt_fs2rmrsxml + xslt_reorder )
sprout_rmrs_morph  = sdl.sprout.SproutModulesTextXml("rmrs-morph.cfg")
xslt_pos_filter    = sdl.xslt.XsltModules("posfilter.xsl", "Chunkie")
sprout_rmrs_lex    = sdl.sprout.SproutModulesXmlXml("rmrs-lex.cfg")
xslt_nodeid_cat    = sdl.xslt.XsltModules("nodeinfo.xsl", "Chunkie")
sprout_rmrs_phrase = sdl.sprout.SproutModulesXmlXml("rmrs-phrase.cfg")
sprout_rmrs_final  = sdl.sprout.SproutModulesXmlXml("rmrs-final.cfg")
xslt_fs2rmrsxml    = sdl.xslt.XsltModules("fs2rmrsxml.xsl")
xslt_reorder       = sdl.xslt.XsltModules("reorderrmrsdtrs.xsl")
```

Figure 6: SDL definition of the SProUT XSLT cascade.

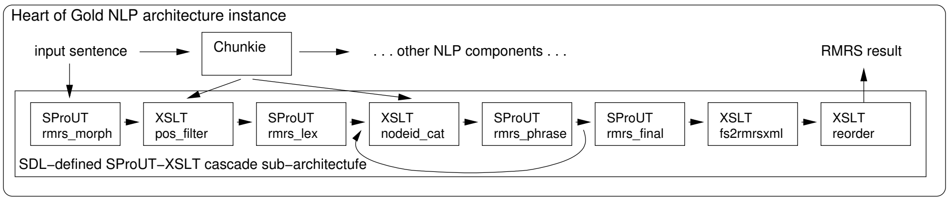

Figure 7: SProUT XSLT cascade in a ‘Heart of Gold’ architecture instance.

4 Semantics Projection Principles for Shallow Grammars

4.1 A Shallow Feature Geometry

The type hierarchy we assume for RMRS construction from shallow grammars specifies expressions as feature structures of type *synsem*, with three main features: the syntactic features NODE and M-SYN, and the semantic feature RMRS (cf. Figure 8.a).

- NODE is used to maintain the constituent information that is needed for structure reparsing: It defines the identifier of the local node (ID) and the mother node’s identifier and category (M-ID, M-CAT). These features are referred to in the rules to restrict rule application to entire constituents.

- M-SYN values convey morpho-syntactic information, namely the category (CAT) and the agreement features person, number, gender, and case (in AGR). In addition, lexical signs store the results of morphological lookup as (typed) inflectional features embedded under M-SYN (cf. section 4.2).

- RMRS, of type *rmrs*, introduces four features: HOOK stores semantic features (a variable and a label) of a sign’s semantics that need to be externalised for semantics composition; RELS is a set containing the elementary predications (EPs) of the local sign; CONS is a set of scope constraints of type *qeq*, with features HI (for the argument positions of quantifiers or other scope-taking items) and LO (for the label of the scoped elementary predication); finally,

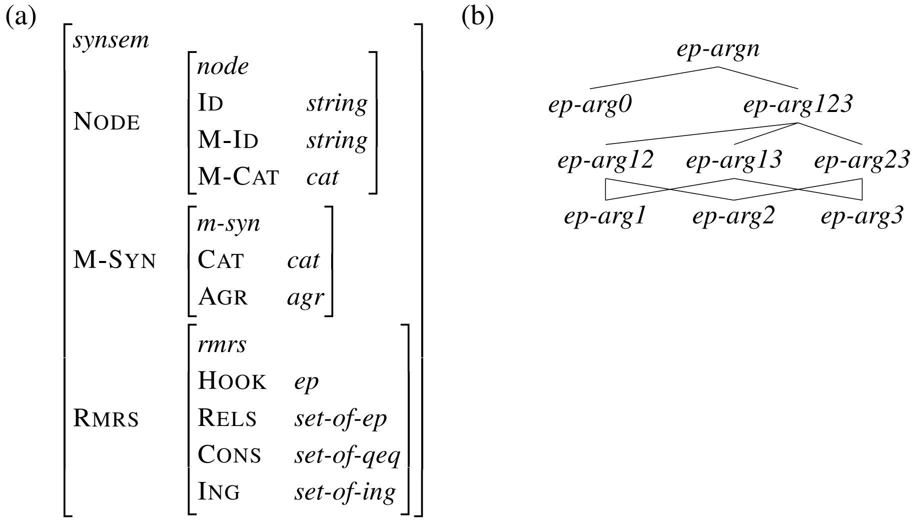

Figure 8: (a) The type *synsem* and (b) the type hierarchy for argument EPs.

> ING is used to encode phrasal grouping of labels, as required for coordination or adjectival modifiers, cf. (Copestake, 2003).

Elementary Predications. The basic units for semantics composition are elementary predications (EPs), of type *ep*. They (minimally) define a label LB (of type *lb*) and a variable VAR. Variables are either of type *hole* (subject to qeq constraints) or of type *individual*, which is again split into *event-var*s with tense and mood information, and *ref(erential)-var*s, carrying PNG information.

We distinguish different subtypes of EPs: *ep-rel* introduces an additional feature REL that specifies the precise semantic relation by means of the lemma, or in terms of general semantic relations (such as *def_rel*, *poss_rel*, etc.);[4] *ep-rstr* and *ep-body* for quantifiers introduce the features RSTR and BODY, respectively. Arguments are encoded as a supertype *ep-argn* with subtypes for underspecified argument types, as shown in Figure 8.b. Note that a general feature name ARGX, introduced by *ep-arg123*, allows us to specify and refer to arguments in a uniform way, irrespective of their (possibly underspecified) argument type. At the lexical level, the ARG0 value of an *ep-arg0* is coreferent with the externalized variable in HOOK in most cases, depending on the lexical class.

Lexical and phrasal types. The type *synsem* is subdivided into *lex* and *phrase* subtypes. While the latter is simply characterized as having a phrasal CAT value, e.g., one of the atomic types *np, vp, pp, ap*, or *s*,[5] the former expands to subtypes corresponding to different word classes. These specify how the PoS-specific

[4] *ep-rel* again expands to several subtypes that correspond to (subclasses of) PoS of lexical items. The PoS-specific subtypes are employed for the definition of PoS-specific semantic conditions in lexical semantics construction rules (see below, Section 4.2).

[5] The category types are determined by the input parser's phrasal category inventory.

morpho-syntactic features (as defined by SProUT's morphological type system in the feature INFL) are mapped to the more general features AGR and CAT in our sign's M-SYN feature.

4.2 RMRS construction from lexical nodes

Interfaces for Morphological Lookup and PoS Filtering. The SProUT system performs morphological lookup on the input string in order to retrieve information about inflectional features (case, number, person, gender, mode, and tense), lemmatization, and PoS.[6] The output structures of morphological processing are (sequences of) TFSs that are based on a hierarchy of (possibly disjunctive) morpho-syntactic types.

Disjunctive types are used for underspecified representation of morphological ambiguities (Krieger and Xu, 2003), instead of atomic disjunctions. Consider, for instance, the German word "Mann" (*man*). This form is ambiguous in that it expresses nominative, accusative or dative case—only genitive ("Mannes") is excluded. Instead of outputting three distinct structures, the morphological component returns one TFS with the underspecified case value *nom_acc_dat*. Rules for morphological agreement, such as the agreement rule in Figure 4, exploit type unification to reduce this ambiguity. E.g., unification of *nom_acc_dat* with *acc_dat* yields the more restricted type *acc_dat*.

As mentioned above, a general rule integrates the purely morphological information provided by morphology lookup (structures of type *morph*) into the M-SYN feature of lexical signs (cf. the feature geometry of Figure 8.a). The rule's LHS matches any structure of the pre-defined type *morph* and introduces it as the M-SYN value of the lexical sign that is defined by the RHS of the rule.

```
morph-lookup :> morph & #1 -> lex & [M-SYN #1].
```

If morphological lookup comes across an unknown word, it returns a TFS not of type *morph*, but of type *token*, with unspecified morphological features. The following rule defines how to handle bare tokens:

```
token-lookup:> token & [SURFACE #1]
               -> lex   & [M-SYN [SURFACE #1,
                                  STEM #1]].
```

This rule acts as a default rule with low priority. Its application is restricted to those parts of the input which fail to match the LHS of the rule morph-lookup. Since there is no morphological information to integrate, the token rule simply enriches the lexical synsem with information about the word stem, which we define as identical to the surface form.

[6] For German, SProUT uses the STTS tagset (Schiller et al., 1999), which supports fine-grained distinctions between word classes. Many of these provide important semantic distinctions, such as different types of pronouns or determiners; e.g., PDS for demonstrative pronouns as in "*This* is great." vs. PDAT for demonstrative determiners ("*This* book is great").

While morphological ambiguities within a given word class (e.g., noun or adjective) are underspecified by means of disjunctive types, the system delivers disjunctive output structures for words that are ambiguous with respect to their PoS. These disjunctions are preserved by the morph-lookup rule applications.

We cut down this type of ambiguity by interfacing the morphological analyses with the categorial information from the original parse tree. We run an XSLT-stylesheet on the rule output, which inserts the category defined by the parser into the CAT feature of the lexical typed feature structures. Since inconsistent structures cannot be matched by any rule, structures with incompatible category specifications are automatically filtered out in the application of the next set of rules.[7]

Moreover, interfacing morphologically enriched lexical structures with the parser's lexical categories provides important word class information for those words that could not be morphologically analysed, and could only be integrated by means of the default token-lookup rule. For these items, we choose the category proposed by the shallow parser for further semantic processing.

Lexical RMRS conditions. Based on the morphologically enriched and PoS-filtered structures, a second rule set introduces lexical RMRS conditions. The individual rules are specific for major PoS lexical classes, again with some special subclasses as provided by the STTS tagset. As an example, we display the rule for common nouns.

```
rmrs-noun:> lex & #lex & [M-SYN [CAT nn,
                              AGR [NUM #num, GEND #gend],
                              INFL infl_noun & [STTS_OPEN_NOUN nn]]]
         -> noun-lex & #lex &
                [RMRS [HOOK ep & [LB #lb, VAR #var],
                       RELS { ep-rel-noun & [LB #lb],
                              ep-arg0 & [LB #lb,
                                         ARG0 ref-var & #var &
                                              [PNG [NUMBER #num,
                                                    GENDER #gend]]] }]].
```

Figure 9: Lexical RMRS conditions (common nouns).

The rule is restricted to apply to lexical signs of category type *nn*, with the appropriate nominal inflectional features under INFL.[8] The RHS of the rule specifies the set of EPs for the lexical sign in RELS: it introduces a noun relation (of type *ep-rel-noun*) and a referential arg0-variable in *ep-arg0*, which is enriched with PNG information from the agreement feature. This variable, and the RMRS label that the two EPs share, constitute the semantic HOOK of the lexical sign.

[7] This presupposes an isomorphic mapping from PoS classes defined in the morphology to PoS classes of the parse tree.

[8] In our implementation, these morphological constraints are factored out as special subtypes of *m-syn* (here *map-morph-nn*). Instead of explicit statement of the morphological constaints, we can thus refer to the appropriate *map-morph-<pos>* type to constrain the application of lexical RMRS rules to specific word classes.

Determination of the concrete contents of the RELS feature at this stage crucially depends on the PoS.[9] The lexical rules for quantifiers, for instance, supply the appropriate EPs of types *ep-rstr* and *ep-body*; possessives introduce an *ep-rel* for the possessive relation, etc.

The output of this level of lexical processing yields RMRSs of the most basic type: Sets of isolated EPs as they can be obtained from a PoS tagger. This bag of "lexical RMRSs" provides the input for the subsequent cascade stages that perform phrasal RMRS composition.

4.3 Content projection principles

RMRS Conditions: Lists vs. Sets. An important issue, in our architecture for semantic composition, is the formal representation of the flat (R)MRS representations. While in theory the values of RELS and CONS are conceived of as sets (or bags), current implementations of typed feature structure formalisms usually do not offer an implementation of sets. MRSs constructed from deep HPSG grammars are therefore represented and processed as (difference) lists.

There have been several approaches to (finite) sets and set unification, some of them extensions to the standard Kasper-Rounds logic for feature structures (e.g., Rounds (1988) or Pollard and Moshier (1990)). Most of them have not been pursued, either due to the the complex nature of the mathematical apparatus, or due to the theoretical and practical complexity (EXPTIME and beyond).

In our approach to semantics construction, independent principles are tailored to specific aspects of semantic composition (e.g., content projection, scoping constraints, or variable binding). Several of these modular principles will apply to the same constituents, and introduce their corresponding semantic constraints. The output structures defined by the individual rules are unified. If the RMRS RELS and CONS features were represented as lists, unification of the output of modular semantics construction rules would in general fail, because list unification is defined by position, and we cannot foresee the relative ordering of semantic predications when different rules apply independently to the same constituent. In our approach, then, we need to represent semantic constraints in RELS, CONS and ING as sets.

In the SProUT system a cheap form of sets (viz. bags) has been implemented that performs *collection*, but not unification of elements into a set (Krieger et al., 2004). That is, the union of two sets $S_1 = \{a_1, b_1\}$ and $S_2 = \{a_2, b_2\}$ will yield the set $S_1 \cup S_2 = \{a_1, b_1, a_2, b_2\}$, whether or not a_1 and a_2 or b_1 and b_2 are unifiable, structurally equivalent, or even identical.

This extension allows us to represent the RMRS features RELS, CONS and ING as sets, and thus to state semantics projection principles in a modular way. The output of the individual semantics projection principles can be unified by set union. To account for the missing unifiability test over set elements, we need to ensure

[9] As a consequence, we obtain PoS-based "default" lexical RMRS conditions for those items that could not be morphologically analysed, but were processed by the token-lookup rule and interfaced with the PoS categories of the parse tree.

that elementary predications are only introduced once. In other words, they need to be sufficiently specified when they are first introduced into the set of semantic constraints. Since RMRS elementary predications are minimal conditions, this can be ensured by appropriate definition of the semantics construction principles.

Structure reparsing for semantic composition. The input to phrasal RMRS composition are sequences of TFSs of type *lex* with isolated lexical RMRS representations, as described in Section 4.2.

The semantic composition of phrases is driven by a general reparsing rule (see Section 3, Figure 3). For each (recursive) application of the phrasal composition rules, the sequence of input TFSs (i.e., the structures built by the previous cascade stage) is enriched with constituency information (ID, M-ID and M-CAT features of NODE) that we extract from the original parse tree by use of an XSLT-stylesheet. By reference to the M-ID features, and given that the system applies longest match, the reparsing rule matches the constituents predicted by the input shallow syntactic parser.[10]

This reparsing rule is now extended with additional constraints to define semantic composition of the matched phrases. This includes principles for the projection of semantic conditions from daughter constituents, as well as principles for variable and argument binding, and scopal constraints.

Basic content projection rule. The content projection rule (Figure 10) assembles the elements of the RMRS RELS, CONS and ING features of all daughter constituents. This is specified by a special collection operator *%{feat}* which refers to the corresponding values *%feat* of the matched constituent phrases. The result structure is defined as the union of the matched *%feat* values.

While a classical list representation would require multiple content projection rules—one for each "arity" of daughter constituents—the set representation enables us to state a single content principle that matches an arbitrary number of daughter constituents. The rule applies to any number of daughter constituents and yields the union of the referenced set-valued features as the semantic value for the mother constituent's feature, here RELS, CONS and ING.

```
cont_proj :> synsem &  [NODE [M-ID #mid],
                        RMRS [RELS %rels, CONS %cons, ING %ing]+
          -> synsem &  [NODE [ID #mid],
                        RMRS [RELS %{rels}, CONS %{cons}, ING %{ing}]].
```

Figure 10: Content projection rule.

The content projection principle is applied to phrasal constituents, and assembles all semantic conditions defined by the daughter constituents to (recursively) define the semantics of phrases. In addition, we define separate principles that conspire

[10] An extended version of the rule in Figure 3 accounts for embedded constituents.

to introduce variable and argument binding as well as scopal constraints that can be defined on the basis of syntactic and morpho-syntactic information.

Variable Binding. Binding of referential variables is defined via the semantic HOOK feature, which is used to externalise variables in compositional semantics construction (see Copestake et al., 2001). As we saw in Section 4.2, in lexical RMRSs the HOOK's variable is in general defined as the internal (ARG0) variable, while in certain cases, such as with adjectives, it is the ARG1 variable that is externalised for referential binding.

The variable binding rule for noun phrases, displayed in Figure 11, refers to the HOOK variables of all daughter constituents of the NP. The rule constrains the referenced variables of all daughters (and all coreferential variables in their lexically defined elementary predications) to be equated. In addition, the rule sets a new HOOK variable for external binding of the phrase, which in the case of noun phrases is identical to the daughter constituents' equated HOOK variables.[11]

```
bind_var :> synsem &  [NODE [M-ID #mid, M-CAT np], RMRS [HOOK [VAR #var ]]]+
            -> phrase &   [NODE [ID #mid], RMRS [HOOK [VAR #var]]].
```

Figure 11: Variable binding.

Scope Constraints. The definition of scope constraints by qeq-constraints in CONS is equally mediated by the HOOK feature. The restrictor argument of quantifiers, for instance, takes scope over the head noun. The corresponding qeq constraint relates the restrictor hole argument of the quantifier and the label of the noun head in a qeq relation. In Figure 12 we display the rule for the introduction of the quantifier's qeq constraint, along with the lexical rule for quantifiers.
In the rmrs-quant rule, the quantifier externalises its ARG0 referential variable as the HOOK's variable (to be used for referential binding), and in addition externalises its main label as the HOOK's LB value. These HOOK features allow us to introduce the quantifier scoping conditions in the q_scope rule. The rule applies to phrases that include a quantifier followed by a noun head. Their respective main labels, #noun_lb and #q_lb, are externalised as HOOK labels, and can thus be used to introduce the corresponding scope conditions into the phrase's RMRS representation: we introduce an elementary predication *ep-rstr* for the quantifier's restrictor argument and a qeq-constraint in CONS, which defines the label of the noun, #noun_lb, to be subordinated to the quantifier's restrictor argument #rstr.

[11] For flat PP structures (as they are typically assumed in shallow parsing, see the tree in Figure 2), we need to separate the binding of referential variables and the definition of the PP's external HOOK variable. Here, the rule restricts the equation of the daughter's variables to the non-prepositional daughters, while the HOOK of the phrase is now defined by the preposition's lexical HOOK variable.

```
bind_var :> prep-lex &  [NODE [M-ID #mid, M-CAT pp], RMRS [HOOK [VAR #prep_var]]
            synsem &  [NODE [M-ID #mid], RMRS [HOOK [VAR #var ]]]*
            synsem &  [NODE [M-ID #mid], RMRS [HOOK [VAR #var ]]]
         -> phrase &   [NODE [ID #mid], RMRS [HOOK [VAR #prep_var]]].
q_scope :> quant-lex &  [NODE [M-ID #mid], RMRS [HOOK [LB #q_lb]]]
           synsem &     [NODE [M-ID #mid]]*
           noun-lex &   [NODE [M-ID #mid], RMRS [HOOK [LB #noun_lb]]]
        -> phrase &     [NODE [ID #mid], RMRS [RELS { ep-rstr & [LB #q_lb, RSTR #rstr] },
                                               CONS { qeq & [HI #rstr, LO #noun_lb] } ]].


     rmrs-quant:> lex & #lex .....
               -> quant-lex & #lex &
                          [RMRS [HOOK [VAR #var, LB #lb]],
                                RELS  {ep-rel & [LB #lb],
                                       ep-arg0 & [LB #lb, VAR #var],
                                       ep-body & [LB #lb, BODY hole] } ]].
```

Figure 12: Scope constraints (quantifiers).

Argument identification and argument binding. Finally, we define semantic composition rules for the binding of arguments. As discussed in Section 3, argument identification may be marked structurally or morphologically.

In our approach, we can define argument binding rules by way of structural constraints for languages like English, as illustrated in Figure 13.[12] The rules identify structural configurations for a VP-external or VP-internal NP, respectively. By way of morpho-syntactic features for active/passive voice (which can be computed by independent morpho-syntactic rules), we can identify or partially restrict the type of argument to be bound.

```
arg-ident-np-vp :> synsem & [M-SYN.CAT np,
                             RMRS.HOOK.VAR #argvar]
                   synsem & [M-SYN [CAT vp, PASSIVE -],
                             RMRS.HOOK.LB #lb]
                -> synsem & [RMRS.RELS  {ep-arg1 & [LB #lb, ARGX #argvar]}].

 arg-ident-v-np :> synsem *
                   synsem & [M-SYN [CAT verb, PASSIVE -],
                             RMRS.HOOK.LB #lb]
                   synsem *
                   synsem & [M-SYN.CAT np,
                             RMRS.HOOK.VAR #argvar]
                   synsem *
                -> synsem & [RMRS.RELS  {ep-arg23 & [LB #lb, ARGX #argvar]}].
```

Figure 13: Structural identification of arguments.

The first rule identifies a VP-external NP in active voice and introduces an elementary predication *ep-arg1* which binds the NP's HOOK variable #argvar as the value of the feature ARGX. The second rule illustrates a case of underspecified argument binding. A VP-internal NP argument (in active voice) may be a direct or indirect argument, depending on the verb's subcategorisation frame. Without lexical infor-

[12] We omit the NODE.M-ID constraints for reparsing here and in the following rules.

mation, we cannot resolve this ambiguity, hence the rule introduces an elementary predication for underspecified argument binding, *ep-arg23*.

For languages with morphological identification of arguments, such as German, we can define argument binding principles that make use of morpho-syntactic constraints, most prominently case. In reparsing we apply agreement rules for morphological disambiguation that lead to maximally resolved case features, in terms of disjunctive types (cf. Section 3, Figure 4).

```
    arg-ident-nom :> synsem∗
                       synsem & [M-SYN [CAT np, AGR.CASE #case],
                                 RMRS.HOOK.VAR #argvar]
                       synsem∗
                       synsem & [M-SYN [CAT verb, PASSIVE -],
                                 RMRS.HOOK.LB #lb]
                       synsem∗
                    -> synsem & [RMRS.RELS {ep-arg1 & [LB #lb, ARGX #argvar]}],
                    where type_eq(#case, nom).

arg-ident-nom-acc :> synsem∗
                       synsem & [M-SYN [CAT np, AGR.CASE #case],
                                 RMRS.HOOK.VAR #argvar]
                       synsem∗
                       synsem & [M-SYN [CAT verb, PASSIVE -],
                                 RMRS.HOOK.LB #lb]
                       synsem∗
                    -> synsem & [RMRS.RELS {ep-arg12 & [LB #lb, ARGX #argvar]}],
                    where type_eq(#case, acc_nom).
```

Figure 14: Morphological identification of arguments.

The rules in Figure 14 apply to sequences of verbs and NP constituents within a phrasal constituent. In the first rule, the case value of the NP constituent is constrained to be of type *nom*, we therefore introduce an EP *ep-arg1* to bind the referential variable of the NP (provided by the HOOK variable #argvar). In the second rule we identify an NP constituent with CASE of type *nom_acc*—the variable is thus bound by way of an underspecified argument binding constraint *ep-arg12*.[13]

Note that the rules make use of a so-called "functional operator" to test for type equality: *type_eq(#case, nom_acc)*. Functional operators are a kind of procedural attachment, which allows us to perform tests that extend the power of type unification. The rules need to distinguish fully disambiguated as opposed to underspecified CASE values in order to introduce the appropriate EP argument type. With type unification, however, we cannot *test* for type equality without *stating* type equality.

That is, if in the first rule we were to constrain the case value of the matched phrase by the specification CASE *nom*, a structure with ambiguous case, such as *nom_acc* could be matched and erroneously disambiguated to *nom*. Vice versa,

[13] Disjunctive versions of these rule take care of alternative head-complement serialisations.

the second rule, if specified to match phrases with ambiguous case, e.g., CASE *nom_acc*, would also apply to fully disambiguated phrases of type *nom*.

The SProUT system enables us to define a functional operator for testing type equality—which in this case can be implemented by way of a simple test on string equality.

As with other semantics construction rules, the rules for argument identification are stated independently for specific arguments or configurations of arguments. The output structures of the individual rules are unified, that is, the corresponding argument identification constraints are assembled in the set-valued RELS feature of the resulting phrases.

Content projection from flat structures. A challenge for principle-based RMRS construction from shallow grammars are their flat syntactic structures. They do not, in general, employ strictly binary structures as assumed in HPSG (see e.g., the semantics construction principles in Flickinger et al. (2003)). Constituents may also contain multiple heads, as with flat PP structures (cf. Figure 2). Finally, chunk parsers do not resolve phrasal attachment, and thus provide discontinuous constituents to be accounted for.

In our reparsing approach for semantics construction, the unification-based pattern matching mechanism of the SProUT system provides elegant means to overcome such difficulties. Independent rules can apply to the same phrases to handle individual aspects of semantics construction. Thus, we can state rules that apply to individual constituents of flat structures, irrespective of the number of phrasal constituents. This enables us to state concise rules for morphological agreement and basic content projection. Similarly, we define independent rules to introduce constraints for scopal relations and argument binding.

For multiple-headed constituents we define special rules with adjusted conditions. For instance, we defined a special bind_var rule for flat PPs (cf. footnote 12) which combines the PP-rule's definition of the phrasal HOOK and the NP-rule's coreference constraints for the binding of referential variables. Due to the modular design of the semantics construction principles and the regular expression-based definition of rules, only minor adjustments are needed to account for flat PPs in the definition of scope constraints, by admitting an optionally preceding preposition.

A more intricate problem are discontinuous structures for complex NP or PP structures as they are delivered by chunk parsers, where phrasal attachments are not resolved. While the basic internal semantic construction rules for NPs and PPs are unaffected by the discontinuous phrasal structures, the argument binding rules must account for the uncertainty of phrasal attachment. Here, we propose to generate in-group conditions that account for possible attachments, along the lines of (Frank, 2003).

Finally, semantics construction from shallow grammars is intrinsically affected by the non-lexicalised nature of these grammars. Due to the lack of lexical subcategorisation information, the principles for semantic composition—especially argument binding—differ significantly from the argument binding principles of deep grammars. While in deep grammars, the binding of arguments can be hard-wired in semantic composition rules, by reference to lexically defined argument "slots" (cf. Copestake et al., 2001), argument binding rules for shallow grammars define constraints on co-occurring constituents to identify their argument status, and generate (potentially underspecified) constraints for argument binding. A natural extension for this type of syntax-semantics interface is the integration of external subcategorisation resources that can be consulted to further constrain the principles for argument binding.

5 Conclusion

We presented an architecture for a constraint-based syntax-semantics interface for RMRS construction from shallow grammars. We proposed a reparsing architecture that permits flexible adaptation to the output of different types of shallow parsers, and argued for a unification-based approach to semantics construction, to account for languages that identify arguments on the basis of morphological constraints. Our reparsing approach permits the definition of modular, interacting semantics construction rules that can be tailored to specific properties of the underlying grammars.

We presented an implementation on the basis of the SProUT processing platform (Drozdzynski et al., 2004; Krieger et al., 2004), a finite-state transduction system that operates on sequences of typed feature structures. The combination of a (cascaded) regular expression-based transduction system with typed feature structure unification turned out to provide a powerful and flexible tool for the definition of complex, but modular semantics construction constraints. In particular, we have argued that the availability of sets as a basic data type is a prerequisite for the implementation of modular semantics construction principles. The usage of a typed feature structure formalism with type inheritance permits concise definition of semantics construction principles.

Compared to the RMRS construction method that Copestake (2003) applies to the English PCFG parser of Carroll and Briscoe (2002), the main features of our approach are (i) argument identification via morphological disambiguation and (ii) definition of modular semantics construction principles in a typed unification formalism. Similar architectures for reparsing have been proposed in earlier work for the generation of LFG f-structures from the output of context-free (PCFG) parsers or treebanks (cf. Frank, 2000; Sadler et al., 2000; Frank et al., 2003b; Cahill et al., 2002; Frank, 2003). Finally, similar ideas that aim at a principled account for RMRS construction from shallow grammars have been independently explored in recent work of Lascarides (2003).

In future work, we will compare our semantics construction principles to the general model of Copestake et al. (2001), a formal framework that was designed for principle-based semantics construction from deep grammars.

References

Becker, M., Drożdżyński, W., Krieger, H.-U., Piskorski, J., Schäfer, U. and Xu, F. 2002. SProUT—Shallow Processing with Unification and Typed Feature Structures. In *Proceedings of the International Conference on Natural Language Processing, ICON-2002*.

Cahill, A., McCarthy, M., van Genabith, J. and Way, A. 2002. Parsing with PCFGs and Automatic F-Structure Annotation. In M. Butt and T.H. King (eds.), *Proceedings of the LFG 2002 Conference*, Athens, Greece: CSLI Online Publications, Stanford, CA.

Callmeier, U., Eisele, A., Schäfer, U. and Siegel, M. 2004. The DeepThought Core Architecture Framework. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC) 2004*, pages 1205–1208, Lisbon, Portugal: European Language Resources Association.

Carpenter, Bob. 1992. *The Logic of Typed Feature Structures*. Tracts in Theoretical Computer Science, Cambridge: Cambridge University Press.

Carroll, C. and Briscoe, E. 2002. High precision extraction of grammatical relations. In *Proceedings of COLING 2002*, pages 134–140.

Copestake, A. 2003. Report on the Design of RMRS. Technical Report D1.1a, University of Cambridge, University of Cambridge, UK., 23 pages.

Copestake, A., Flickinger, D., Sag, I. and Pollard, C. 2003. Minimal Recursion Semantics, ms.

Copestake, A., Lascarides, A. and Flickinger, D. 2001. An Algebra for Semantic Construction in Constraint-based Grammars. In *Proceedings of the ACL 2001*, Toulouse, France.

Daum, M., Foth, K.A. and Menzel, W. 2003. Constraint-based Integration of Deep and Shallow Parsing Techniques. In *Proceedings of EACL 2003*, Budapest, Hungary.

Drozdzynski, W., Krieger, H.-U., Piskorski, J., Schäfer, U. and Xu, F. 2004. Shallow Processing with Unification and Typed Feature Structures — Foundations and Applications. *Künstliche Intelligenz* 1, 17–23.

Egg, M., Koller, A. and Niehren, J. 2001. The Constraint Language for Lambda Structures. *Journal of Logic, Language, and Information* 10, 457–485.

Flickinger, D., Bender, E. M. and Oepen, S. 2003. MRS in the LinGO Grammar Matrix: A Practical User's Guide. Technical Report, Deep Thought Project Deliverable 3.5.

Frank, A. 2000. Automatic F-structure Annotation of Treebank Trees. In M. Butt and T.H. King (eds.), *Proceedings of the LFG00 Conference*, CSLI Online Publications, Stanford, CA.

Frank, A. 2003. Projecting LFG F-Structures from Chunks – or (Non-) Configurationality from a Different Viewpoint. In M. Butt and T.H. King (eds.), *Proceedings of the LFG 2003 Conference*, pages 217–237, Albany, New York: CSLI Publications.

Frank, A., Becker, M., Crysmann, B., Kiefer, B. and Schäfer, U. 2003a. Integrated Shallow and Deep Parsing: ToPP meets HPSG. In *Proceedings of the ACL 2003*, pages 104–111, Sapporo, Japan.

Frank, A., Sadler, L., van Genabith, J. and Way, A. 2003b. From Treebank Resources to LFG F-Structures. Automatic F-Structure Annotation of Treebank Trees and CFGs Extracted from Treebanks. In A. Abeille (ed.), *Building and Using Syntactically Annotated Corpora*, The Netherlands: Kluwer Academic Publishers.

Hinrichs, E. and Trushkina, J. 2002. Getting a Grip on Morphological Disambiguation. In S. Busemann (ed.), *Proceedings of KONVENS 2000*, Saarbrücken, Germany.

Krieger, H.-U. 2003. SDL—A Description Language for Specifying NLP Systems. In *Proceedings of the 3rd AMAST Workshop on Algebraic Methods in Language Processing, AMiLP-3*.

Krieger, H.-U., Drozdzynski, W., Piskorski, J., Schäfer, U. and Xu, F. 2004. A Bag of Useful Techniques for Unification-Based Finite-State Transducers. In Ernst Buchberger (ed.), *Proceedings of 7th KONVENS*.

Krieger, H.-U. and Xu, F. 2003. A type-driven method for compacting MMorph resources. In *Proceedings of RANLP 2003*, pages 220–224.

Lascarides, A. 2003. Robust Construction, manuscript, University of Edinburgh.

Müller, F. H. 2004. Annotating Grammatical Functions for German Using Finite-Stage Cascades. In *Proceedings of COLING 2004*, pages 268–274, Geneva, Switzerland.

Pollard, Carl J. and Moshier, M. Drew. 1990. Unifying Partial Descriptions of Sets. In P. Hanson (ed.), *Information, Language, and Cognition. Vol. 1 of Vancouver Studies in Cognitive Science*, pages 285–322, University of British Columbia Press.

Reyle, U. 1993. Dealing with Ambiguities by Underspecification: Construction, Representation and Deduction. *Journal of Semantics* 10(2), 123–179.

Rounds, William C. 1988. Set Values for Unification-Based Grammar Formalisms and Logic Programming. Technical Report CSLI-88-129, Center for the Study of Language and Information.

Sadler, L., van Genabith, J. and Way, A. 2000. Automatic F-Structure Annotation from the AP Treebank. In M. Butt and T.H. King (eds.), *Proceedings of the LFG00 Conference*, University of California, Berkley, CSLI Online Publications, Stanford, CA.

Schäfer, U. 2004. Using XSLT for the Integration of Deep and Shallow Natural Language Processing Components. In *Proceedings of the ESSLLI 2004 workshop on Combining Shallow and Deep Processing for NLP*, pages 31–40, Nancy, France.

Schiller, A. Teufel, S. and Stöckert, C.İ999. Guidelines für das Tagging deutscher Textcorpora mit STTS. Technical Report, University of Stuttgart and Tübingen.

Skut, W. and Brants, T. 1998. Chunk tagger: statistical recognition of noun phrases. In *ESSLLI-1998 Workshop on Automated Acquisition of Syntax and Parsing*, Saarbrücken.