The TIGER 700 RMRS Bank: RMRS Construction from Dependencies

Kathrin Spreyer
Computational Linguistics Department
Saarland University
66041 Saarbrücken, Germany
kathrins@coli.uni-sb.de

Anette Frank
Language Technology Lab
DFKI GmbH
66123 Saarbrücken, Germany
frank@dfki.de

Abstract

We present a treebank conversion method by which we construct an RMRS bank for HPSG parser evaluation from the TIGER Dependency Bank. Our method effectively performs automatic RMRS semantics construction from functional dependencies, following the semantic algebra of (Copestake et al., 2001). We present the semantics construction mechanism, and focus on some special phenomena. Automatic conversion is followed by manual validation. First evaluation results yield high precision of the automatic semantics construction rules.

1 Introduction

Treebanks are under development for many languages. They are successfully exploited for the induction of treebank grammars, training of stochastic parsers, and for evaluating and benchmarking competitive parsing and grammar models. While parser evaluation against treebanks is most natural for treebank-derived grammars, it is extremely difficult for hand-crafted grammars that represent higher-level functional or semantic information, such as LFG or HPSG grammars.

In a recent joint initiative, the TIGER project provides dependency-based treebank representations for German, on the basis of the TIGER treebank (Brants et al., 2002). (Forst, 2003) applied treebank conversion methods to the TIGER treebank, to derive an f-structure bank for stochastic training and evaluation of a German LFG parser. A more theory-neutral dependency representation is currently derived from this TIGER-LFG treebank for cross-framework parser evaluation (Forst et al., 2004). However, while Penn-treebank style grammars and LFG analyses are relatively close to dependency representations, the output of HPSG parsing is difficult to match against such structures. HPSG analyses do not come with an explicit representation of functional structure, but directly encode semantic structures, in terms of (Robust) Minimal Recursion Semantics (henceforth (R)MRS.[1] This leaves a gap to be bridged in terms of the encoding of arguments vs. adjuncts, the representation of special constructions like relative clauses, and not least, the representation of quantifiers and their (underspecified) scoping relations.

In order to bridge this gap, we construct an RMRS "treebank" from a subset of the TIGER Dependency Bank (Forst et al., 2004), which can serve as a gold standard for HPSG parsing for evaluation, and for training of stochastic HPSG grammar models. In contrast to treebanks constructed from analyses of hand-crafted grammars, our treebank con-

[1] RMRS (Copestake, 2003) is a formalism for partial semantic representation that is derived from MRS (Copestake et al., 2005). It is designed for the integration of semantic representations produced by NLP components of different degrees of partiality and depth, ranging from chunk parsers and PCFGs to deep HPSG grammars with (R)MRS output.

version approach yields a standard for comparative parser evaluation where the upper bound for coverage is defined by the corpus (here, German newspaper text), not by the grammar.

Our method for treebank conversion effectively performs priniciple-based (R)MRS semantics construction from LFG-based dependency representations, which can be extended to a general parsing architecture for (R)MRS construction from LFG f-structures.

The remainder of this paper is organised as follows. Section 2 introduces the input dependency representations provided by the TIGER Dependency Bank, and describes the main features of the term rewriting machinery we use for treebank conversion, i.e., RMRS semantics construction from dependency structures. Section 3 presents the core of the semantics construction process. We show how to adapt the construction principles of the semantic algebra of (Copestake et al., 2001) to RMRS construction from dependencies in a rewrite scenario, and discuss the treatment of some special phenomena, such as verbal complementation, coordination and modification. Section 4 reports on the treebank construction methodology, with first results of quality control. Section 5 concludes.

2 From TIGER Dependency Bank to TIGER RMRS Bank

2.1 The TIGER Dependency Bank

The input to the treebank conversion process consists of dependency representations of the TIGER Dependency Bank (TIGER-DB). The TIGER-DB has been derived semi-automatically from (a subset of) the TIGER-LFG Bank of (Forst, 2003), which is in turn derived from the TIGER treebank. The dependency format is similar to the Parc 700 Dependency Bank (King et al., 2003). It abstracts away from constituency in order to remain as theory-neutral as possible. So-called dependency triples are sets of two-place predicates that encode grammatical relations, the arguments representing the head of the dependency and the dependent, respectively. The

```
sb(müssen~0, Museum~1)
oc_inf(müssen~0, weichen~3)
mood(müssen~0, ind)
tense(müssen~0, pres)
mod(Museum~1, privat~1001)
cmpd_lemma(Museum~1, Privatmuseum)
case(Museum~1, nom)
gend(Museum~1, neut)
num(Museum~1, sg)
sb(weichen~3, Museum~1)
```

Figure 1: TIGER dependency representation of sentence #8595: *Privatmuseum muss weichen* – Private museum deemed to vanish.

triples further retain a number of morphological features from the LFG representations, such as agreement features or tense information. Figure 1 displays an example.

For the purpose of RMRS construction, the triples format has advantages and disadvantages. The LFG-derived dependencies offer all the advantages of a functional as opposed to a constituent-based representation. This representation already filters out the semantically inappropriate status of auxiliaries as heads; their contribution is encoded by features such as `perf` or `fut`, which can be directly translated into features of semantic event variables. Most importantly, the triples localise dependencies which are not locally realised in phrase structure (as in long-distance constructions), so that there is no need for additional mechanisms to identify the arguments of a governing predicate. Moreover, the dependency representation format is to a large extent uniform across languages, in contrast to phrase-structural encoding. Therefore, the dependency-based semantics construction mechanism can be quickly ported to other languages.

The challenges we face mainly concern a lack of specific types of phrase structure information that are crucial for RMRS composition. Linear precedence, e.g., plays a crucial role when it comes to multiple modification or coordination. Yet, it is possible to reconstruct the surface order from the indices attached to the PRED values in the triples. Part-of-speech information, which is useful to trigger different types of semantics construction rules, can be induced from the presence or absence of

certain morphological features, yet to a limited extent.

For our current purpose of treebank conversion, we are dependent on the specific input format of the TIGER-DB, while in a more general parsing context, one could ensure that missing information of this type is included in the input to semantics construction.

2.2 Treebank Conversion

Similar to the TIGER to TIGER-DB conversion of (Forst, 2003; Forst et al., 2004), we are using the term rewriting system of (Crouch, 2005) for treebank conversion. Originally designed for machine translation, the system is a powerful rewriting tool that has been applied to other tasks, such as frame semantics construction (Frank and Erk, 2004), or induction of knowledge representations (Crouch, 2005).

The input to the system consists of a set of facts in a prolog-like term representation. The rewrite rules refer to these facts in the left-hand side (LHS), either conjunctively (marked by '`,`') or disjunctively (marked by '|'). Expressions on the LHS may be negated (by prefix '`-`'), thereby encoding negative constraints for matching. A rule applies if and only if all facts specified on the LHS are satisfied by the input set of facts. The right-hand side (RHS) of a rewrite rule defines a conjunction of facts which are added to the input set of facts if the rule applies. The system further allows the user to specify whether a matched fact will be consumed (i. e., removed from the set of facts) or whether it will be retained in the output set of facts (marked by prefix '`+`').[2]

The system offers powerful rule encoding facilities in terms of macros and templates. Macros are parameterized patterns of (possibly disjunctive) facts; templates are parameterized abstractions over entire (disjunctive) rule applications. These abstraction means help the user to define rules in a perspicious and modular way, and significantly enhance the maintainability of complex rule sets.

The processing of rules is *strictly ordered*. The rules are applied in the order of textual appearance. Each rule is tested against the current input set of facts and, if it matches, produces an output set of facts that provides the input to the next rule in sequence. Each rule applies concurrently to all distinct sets of matching facts, i.e. it performs parallel application in case of alternative matching facts.

3 RMRS Construction from Dependencies

Within the framework of HPSG, every lexical item defines a complete (R)MRS structure. The semantic representation of a phrase is defined as the assembly and combination of the RMRSs of its daughters, according to semantic constraints, which apply in parallel with syntactic constraints. In each composition step, the RMRSs of the daughters are combined according to semantic composition rules that define the semantic representation of the phrase, cf. (Copestake et al., 2005). Following the scaffolding of the syntactic structure in this way finally yields the semantics of the sentence.

For the present task, the input to semantics construction is a dependency structure. As established by work on Glue Semantics (Dalrymple, 1999), semantics construction from dependency structures can in similar ways proceed recursively, to deliver a semantic projection of the sentence. Note, however, that the resource-based approach of Glue Semantics leads to alternative derivations in case of scope ambiguities, whereas RMRS targets an underspecified semantic representation.

For (R)MRS construction from dependencies we follow the algebra for semantics composition in (Copestake et al., 2001). In HPSG implementations of this algebra, composition is triggered by phrasal configurations. Yet, the algebra is neutral with regard to the syntactic representation, and can be transposed to composition on the basis of dependency relations, much alike the Glue framework.

However, the rewriting system we are using is not suited for a typical recursive application

[2] The system additionally features *optional rules* ('`?=>`'), as opposed to deterministic rewriting ('`==>`'). However, given that the input structures for RMRS construction are disambiguated, and since our target structures are underspecified semantic structures, we can define the semantics deterministically.

scheme: the rules are strictly ordered, and each rule simultaneously applies to all facts that satisfy the constraints in the LHS. That is, the RMRS composition cannot recursively follow the composition of dependents in the input structure. In section 3.2 we present a design of RMRS that is suited for this concurrent application scheme. Before, we briefly sketch the semantic algebra.

3.1 An Algebra for Semantic Construction

(Copestake et al., 2001) define a semantic entity as a 5-tuple $\langle s_1, s_2, s_3, s_4, s_5 \rangle$ such that s_1 is a *hook*, s_2 is a (possibly empty) set of *holes*, s_3 and s_4 are bags of *Elementary Predications* (EPs) and *handle constraints*, respectively, and s_5 is a set of equalities holding between variables. The hook is understood to represent the externalised part of the semantic entity as a pair of a handle and an index (a variable). It is used for reference in composition: Hooks of semantic arguments fill holes (or slots) of functors. Holes, in turn, record gaps in a semantic representation which remain to be filled. They, too, are pairs of a handle and an index; furthermore, holes are labelled with the grammatical function they bear syntactically. That is, the labels on holes serve two purposes: They help determine the appropriate operation of composition (see below), and they link the semantics to syntax.[3]

EPs (predicate applications) represent the binding of argument variables to their predicators. An EP $h : r(a_1, \ldots, a_n, sa_1, \ldots, sa_m)$ consists of the EP's handle (or label) h, a relation r, and a list of zero or more variable arguments $a_1, \ldots, a_n$, followed by zero or more scopal arguments $sa_1, \ldots, sa_m$ (denoting handles) of the relation. Finally, the bag of handle constraints (HCONS) contains conditions which (partially) specify the relations between scopal arguments and their scope, i.e. between the scopal argument and the handles that may fill the hole.

The operators of semantic composition $\mathrm{op}_{l_1}, \ldots, \mathrm{op}_{l_k}$ are drawn from $\Sigma \times \Sigma \rightarrow \Sigma$, where Σ is the set of all semantic entities, and $l_1, \ldots, l_k$ correspond to the labels on holes: An operator op_{l_i} defines the composition of a semantic head which has a hole labelled l_i with the argument filling that hole as follows: The result of $\mathrm{op}_{l_i}(a_1, a_2)$ is undefined if a_2 has no hole labelled l_i, otherwise:

1. $\mathrm{hook}(\mathrm{op}_{l_i}(a_1, a_2)) = \mathrm{hook}(a_2)$;
2. $\mathrm{holes}_{l'}(\mathrm{op}_{l_i}(a_1, a_2)) = \mathrm{holes}_{l'}(a_1) \cup \mathrm{holes}_{l'}(a_2)$ for all labels $l' \neq l_i$;
3. $\mathrm{eps}(\mathrm{op}_{l_i}(a_1, a_2)) = \mathrm{eps}(a_1) \bigoplus \mathrm{eps}(a_2)$;
4. $\mathrm{eqs}(\mathrm{op}_{l_i}(a_1, a_2)) = \mathrm{Tr}(\mathrm{eqs}(a_1) \cup \mathrm{eqs}(a_2) \cup \{\mathrm{hook}(a_1) = \mathrm{hole}_{l_i}(a_2)\})$; where Tr stands for the transitive closure.

3.2 RMRS Design

As mentioned earlier, the concurrent nature of rule application makes it impossible to proceed recursively in a scaffolding way, inherent to tree-based analyses, since the rules apply simultaneously to all structures. RMRS construction is therefore designed around one designated "global" RMRS. Instead of projecting and accumulating RMRS constraints step-wise by recursive composition, we directly insert the meaning descriptions into a single *global RMRS*. Otherwise, composition strictly follows the semantic operations of the algebra of (Copestake et al., 2001): the composition rules only refer to the hook and slots of functors and arguments, to achieve the binding of argument variables and the encoding of scope constraints.

Global and Lexical RMRSs. The *global RMRS* features a top handle (TOP, usually the label of the matrix proposition), sets of EPs (RELS) and handle constraints (HCONS), respectively, as described in the algebra, and a set of ING constraints.[4]

[3] (Copestake et al., 2001) mention a third feature to be included in the hook as an externally visible variable, which they instantiate with the index of the controlled subject in equi constructions and which is also used to implement the semantics of predicative modification. However, this feature is not crucial given that the underlying syntactic structures represent dependencies rather than immediate dominance relations, and therefore make non-local information available locally. Likewise, the dependency scenario does not necessitate that modifiers externalise their ARG1 argument position (see section 3.3.3).

[4] Whenever two labels are related via an ING (in-group) constraint, they can be understood to be con-

```
+pred(X,Pred),-mo(_,X), -spec(_,X),
+'s::'(X,SemX), +hook(SemX,Hook)
==> lb(Hook,Lb), var(Hook,Var)
  && add_ep(Lb,ep_rel,rel,Pred)
  && add_ep(Lb,ep_arg0,arg0,Var).
```

lexical RMRS: $\left[\textsc{Hook}\ \left[\begin{array}{ll}\textsc{Lb} & \boxed{\text{Lb}}\\ \textsc{Var} & \boxed{\text{Var}}\end{array}\right]\right]$

global RMRS: $\left[\begin{array}{ll}\textsc{Rels} & \left\{\ldots, \left[\begin{array}{ll}\textit{\_Pred\_n} & \\ \textsc{Lb} & \boxed{\text{Lb}}\\ \textsc{Arg0} & \boxed{\text{Var}}\end{array}\right], \ldots\right\}\\ \textsc{Hcons} & \{\ldots\}\\ \textsc{Ing} & \{\ldots\}\end{array}\right]$

Figure 2: A rule for nominals (top) with resulting lexical and global RMRS (bottom).

In addition, every predicate in the dependency structure projects a *lexical RMRS*. Lexical RMRSs are semantic entities which consist of only a hook (i.e. a label and a variable), that makes the entity available for reference by subsequent (composition) rules, whereas the basic semantic content (which is determined on the basis of the predicate's category, and comprises, at least, EPs for the relation and the ARG0)[5] is uniformly maintained in the bags of the global RMRS, yet still anchored to the lexical hook labels and variables.

Figure 2 shows an example of a lexical RMRS with its links to the global RMRS, and a simplified version of the corresponding rule: The rule applies to predicates, i.e. `pred` features, with a value `Pred`. It introduces the lexical RMRS, i.e., the hook's label and variable, and adds the predicate's basic semantic content to the global RMRS, here the relation represented by `Pred` and the ARG0 variable, which is co-referent with the hook's variable.

Composition. The semantic composition of arguments and functors is driven by the predicate `arg(Fctor,N,Arg)`, where `N` encodes the argument position, `Fctor` and `Arg` are indices of functor and argument, respectively.[6] We interpret the `arg`-predicate as a slot/hole of the functor, such that the binding of the argument to the functor comes down to filling the hole, in the sense of the algebra described above: This is steered by the previously defined hooks of the two semantic entities, in that the matching rule introduces an EP with an attribute ARGN that is anchored to the externalised label in the functor's hook. The value of the attribute ARGN is the hook variable or hook label of the argument, depending on the category. A slightly more complicated example is shown in Figure 3, it involves the introduction of an additional proposition and a scope constraint. This rule performs the composition of a declarative finite clausal object (`oc_fin`) with its verbal head. It assigns a proposition relation as the value of the verb's ARG2, which in turn has an ARG0 that takes scope over the hook label of the matrix verb in the object clause.

In general, composition does not depend on the order of rule applications. That is, the fact that the system performs concurrent rule

```
+arg(X,2,Arg), +g_f(Arg,'oc_fin'),
+comp_form(Arg,dass)
get_lb(X,LbX), get_lb(Arg,LbA)
==> sort(Lb,h), sort(LbP,h)
  && add_ep(LbX,ep_arg2,argx,LbP)
  && add_ep(LbP,ep_rel,rel,'prpstn_m_rel')
  && add_ep(LbP,ep_arg0,arg0,Lb)
  && add_qeq(Lb,LbA).
```

lexical RMRSs:

X: $\left[\textsc{Hook}\ \left[\textsc{Lb}\ \boxed{\text{LbX}}\right]\right]$ Arg: $\left[\textsc{Hook}\ \left[\textsc{Lb}\ \boxed{\text{LbA}}\right]\right]$

global RMRS:

$$\left[\begin{array}{ll}\textsc{Rels} & \left\{.., \left[\begin{array}{ll}\textit{\_X\_v} & \\ \textsc{Lb} & \boxed{\text{LbX}}\\ \textsc{Arg2} & \boxed{\text{LbP}}\end{array}\right], \left[\begin{array}{ll}\textit{prpstn\_m\_rel} & \\ \textsc{Lb} & \boxed{\text{LbP}}\\ \textsc{Arg0} & \boxed{\text{Lb}}\end{array}\right], \left[\begin{array}{ll}\textit{\_Arg\_v} & \\ \textsc{Lb} & \boxed{\text{LbA}}\end{array}\right], ..\right\}\\ \textsc{Hcons} & \{\ldots, \boxed{\text{Lb}}\ \text{qeq}\ \boxed{\text{LbA}}, \ldots\}\end{array}\right]$$

Figure 3: Sample argument binding rule triggered by `arg(X,2,Arg)` (top), referred lexical RMRSs and resulting global RMRS (bottom).

joined. This is relevant, e.g., for intersective modification, since a quantifier that outscopes the modified noun must also take scope over the modifier.

[5]The category information required to define the concrete basic semantics is not explicit in the dependencies, but is induced from the grammatical function borne by the predicate, as well as the presence or absence of certain morphological features (section 2.1).

[6]The `arg` predicates are introduced by a set of preprocessing rules which reconstruct the argument structure by referring to the local grammatical functions of a predicate and testing for (morphological) features typically borne by non-arguments. E.g., `pron_type(_,expl)` identifies an expletive pronoun.

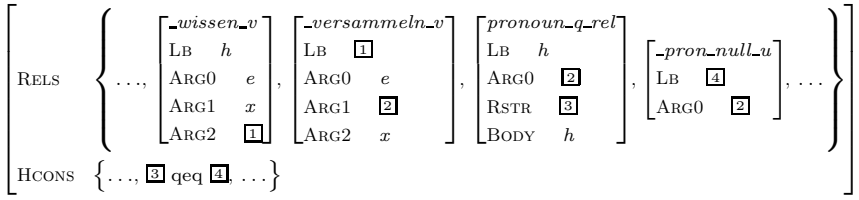


Figure 4: Control analysis with unspecified coreference in *[...], als so gut wie er kaum ein anderer die Studentenmassen [...] zu versammeln wußte.* – [...] when hardly anybody knew how to rally the crowd of students [...] as well as he did. (from corpus sentence # 8074).

applications in a cascaded rule set is not problematic for semantics construction. Though, we have to ensure that every partial structure is assigned a hook, prior to the application of composition rules. This is ensured by stating the rules for lexical RMRSs first.

Scope constraints. By introducing handle constraints, we define restrictions on the possible scoped readings. This is achieved by gradually adding qeq relations to the global HCONS set. Typically, this constraint relates a handle argument of a scopal element, e.g. a quantifier, and the label of the outscoped element. However, we cannot always fully predict the interaction among several scoping elements. This is the case, inter alia, for the modification of verbs by more than one scopal adverb. This ambiguity is modeled by means of a UDRT-style underspecification, that is, we leave the scope among the modifiers unspecified, but restrict each to outscope the verb handle.[7]

3.3 Selected Phenomena

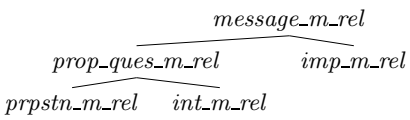
3.3.1 Verbal complements.

The treebank distinguishes three kinds of verbal complements: infinitival phrases govered by a raising verb or by a control verb, and finite clausal arguments.

Infinitival complements. Raising verbs do not assign an ARG1, and the infinitival argument is bound via an additional proposition which fills the ARG2 position of the governor. A handle constraint requires the proposition to take scope over the label of the infinitive. Modal verbs lend themselves most naturally to the same analysis, by virtue of identical annotation in the dependency triples.

The implementation of RMRS for equi constructions relies on external lexicon resources, since the underlying dependency structures do not encode the coreference between the controlled subject and the external controller. Instead, the controlee is annotated as a null pronoun. In order to differentiate subject from object control, we enrich the transfer input with a list of static facts `s_control(Pred)` and `o_control(Pred)`, respectively, which we extracted from the German HPSG grammar (Crysmann, 2003). The rules refer to these facts, and establish the appropriate bindings. If no information about coreference is available (due to sparse lexical data), the controlled subject appears in the RMRS as an unbound pronoun, as assumed in the syntactic structure. This is shown in Fig. 4. In the manual correction phase, these cases are corrected in the output RMRS, by introducing the missing control relation.

Finite complements. For finite clausal complements we assume the basic analysis illustrated in section 3.2. But finite clauses are not necessarily declarative, they can also have interrogative meaning. In RMRS, this distinction is typically drawn in a type hierarchy, of which we assume a simplified version:



German embedded clauses are usually marked by one of the complementizers *dass* (that)

[7] This is in accordance with the German HPSG grammar of (Crysmann, 2003), and will also be adapted in the ERG (p.c. D. Flickinger).

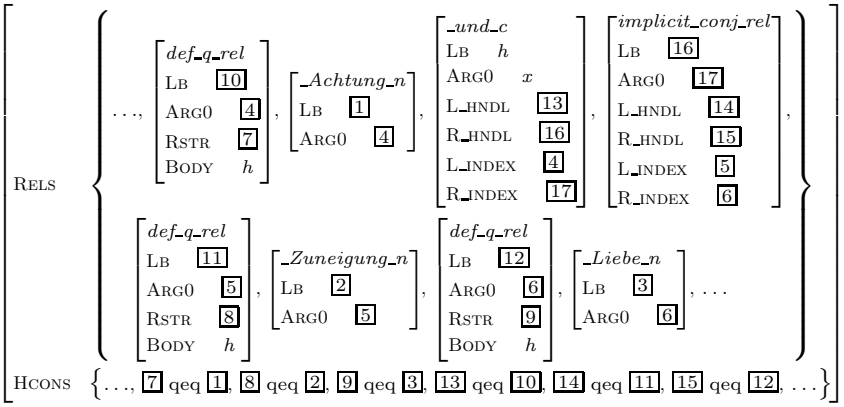


Figure 5: RMRS for the coordinate NP *ihre Achtung, ihre Zuneigung und Liebe* – their esteem, their affection and love (from corpus sentence # 8345).

or *ob* (whether), in initial position, but may occur without it, though less frequently. If a complementizer is present, this is recorded as `comp_form(_,dass)` (resp. `comp_form(_,ob)`), and we can fully determine the kind of message relation from its lexical form, i.e., *prpstn_m_rel* for declarative and *int_m_rel* for interrogative ones. In the absence of an overt complementizer, we could introduce the underspecified type *prop_ques_m_rel*, but rather chose to use a default rule for the declarative reading *prpstn_m_rel*, which occurs far more often. This reduces the manual correction effort.

3.3.2 Coordination

The HPSG analysis of coordinate structures takes the form of a binary, right-branching structure. Since semantics construction in HPSG proceeds along this tree, an RMRS for a coordinate phrase likewise mirrors the recursive organisation of conjuncts in the syntax. Each partial coordination introduces an *implicit_conj_rel*, while the meaning contributed by the lexical conjunction is conveyed in the EP which spans the entire coordination.

By contrast, the dependency structures preserve the flat LFG-analysis of coordination as a set of conjuncts. To overcome this discrepancy between source and target structures, we define specialised rules that mimic recursion in that they process the conjuncts from right to left, two at a time, thereby building the desired, binary-structure semantics for the coordination. Fig. 5 shows a sample output RMRS for coordinated NPs.[8] Note that we posit the L/R_HNDL handle arguments to outscope each label that takes scope over the noun. This accounts for scope ambiguities among quantifiers and scopal adjectives.

3.3.3 Recursive Modification

The algebra of (Copestake et al., 2001) defines modifiers to externalise the variable of the ARG1. This, however, runs into problems when a construction needs to incorporate the inherent event variable (ARG0) of a modifier as an argument, as e.g. in recursive modification. In these cases, the ARG0 variable is not accessible as a hook for composition.

In contrast, we identify the hook variable of modifiers with their ARG0 variable. This enables a uniform account of recursive intersective modification, since the inherent variable is legitimatly accessible via the hook, whereas the ARG1—like any other argument—is bound in a slot-filling operation.[9] The corresponding rule and an example output RMRS are displayed in Fig. 6: Whenever the dependency relation `mo` is encountered, no matter what the exact `pred` value, the semantics contributed by the head of the

[8] The semantic contribution of the possessive pronouns has been neglected for ease of exposition.

[9] Similarly, this treatment of modification correctly accounts for modification in coordination structures, as in the NP *ihrer munteren und farbenfreudigen Inszenierung* – of her lively and colourful production (from corpus sentence # 9821).

```
+mo(X,M), +pred(M,Pred),
-scopal(Pred),
+'s::'(M,SemM), +hook(SemM,Hook),
+lb(Hook,LbM),
get_var(X,VarX), get_lb(X,LbX)
==> var(Hook,VarM)
&& add_ep(LbM,ep_rel,rel,Pred)
&& add_ep(LbM,ep_arg0,arg0,VarM)
&& add_ep(LbM,ep_arg1,argx,VarX)
&& add_ing(LbM,LbX).
```

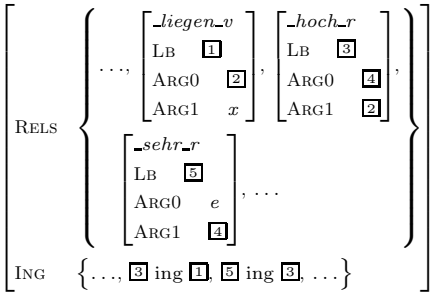


Figure 6: Rule defining the lexical RMRS for modifiers (top), resulting global RMRS for the recursive modification in *liege [...] sehr hoch* – [...] is at a very high level (from corpus sentence # 8893).

dependency can be unambiguously identified as the argument of the semantic head. In fact, given that modifiers are in this way locally annotated as `mo` dependents in the triples, we can bind the ARG1 already when defining the lexical RMRS of the modifier.

4 The TIGER 700 RMRS Bank

4.1 Design and methodology

Treebank Design. Our aim is to make available manually validated RMRS structures for 700 sentences of the TIGER-DB. Since the underlying data is contiguous newspaper text, we chose to select a block of consecutive sentences instead of a random sample. In this way, the treebank can be further extended by annotation of intersentential phenomena, such as co-reference or discourse relations.

However, we have to accommodate for gaps, due to sentences for which there are reasonable functional-syntactic, but (currently) no sound semantic analyses. This problem arises for sentences involving, e.g., elliptical constructions, or else ungrammatical or fragmented sentences. We will include, but explicitly mark such sentences for which we can only obtain partial, but no fully sound semantic analyses. We will correspondingly extend the annotation set to yield a total of 700 correctly annotated sentences.

The composition rules are designed to record their application by way of rule-specific identifiers. These may serve as a filtering means in case the analysis of certain phenomena as assumed in the treebank is incompatible with the grammar to be evaluated.

Quality Control. For compilation of a manually controlled RMRS bank, we implemented a cascaded approach for quality control, with a feedback loop between (i) and (ii):

(i) Manual sample-based error-detection. We are using the application markers of specific construction rules to select sample RMRSs for phenomenon-based inspection, as well as random sampling, in order to detect problems that can be corrected by adjustments of the automatic conversion procedure.

(ii) Adjustment of conversion rules. The construction rules are modified to adjust errors detected in the automatic conversion process. Errors that cannot be covered by general rules need to be manually corrected in (iii).

(iii) Manual control. Finally, we perform manual control and correction of errors that cannot be covered by automatic RMRS construction. Here, we mark and separate the phenomena that are not covered by the state-of-the-art in RMRS-based semantic theory.

Viewing and editing support. The inspection of RMRSs is supported by converting the underlying XML format to HTML. RMRSs can thus be comfortably viewed in a browser, with highlighting of coreferences, display of agreement features, and links of EPs to the surface forms they originated from.

Correction is supported by an XSLT-based interactive editing tool. It enables the user to specify which EPs, arguments or constraints are to be added/removed. With each change, the HTML representation is updated, so that the result is immediately visible for verification. The tool features a simple mechanism for version maintenance and retrieval, and

	abs #	in %	avg # of corrections/sent.
validated	700	100	2.24
odd	28	4	
fully perfect	281	40.14	0
corrected	419	59.86	3.75
<5 corrections	601	85.86	0.96

	abs #	in %	avg # of corrections/sent.
validated	100	100	1.3
odd	5	5	
fully perfect	68	68	0
corrected	32	32	4.2
<5 corrections	88	88	0.44

Table 1: Evaluation of current data set for complete correction (top) and for correction ignoring part-of-speech (bottom).

separate storage for fully validated structures.

4.2 First Results

The transfer grammar comprises 74 rewrite rules for converting dependency structures to RMRS, plus 34 macros and templates.

In a first validation experiment on the basis of 100 structures, we classified 20% of the RMRSs as involving errors that can be captured by adjustments of the automatic conversion rules (see step (ii) above), while 59% were fully correct.[10]

After improvement of the rules we evaluated the quality of the automatic construction procedure by validating the 700 sentences of the treebank. Average counts for this sample are 15.57 tokens/sentence, 15.92 dependencies/sentence. Table 1 (top) summarises the results. Of the 700 structures, 4% contained phenomena which we do not analyse at all. 40% required no correction at all. For the 59% that needed manual correction, the average count of units to be corrected per sentence was 3.75. The number of RMRSs that needed less than the average of corrections was 601, i.e. 85.86%. The time needed for inspection and correction was 5 mins 12 secs/sentence, calculated on the entire data set.

Error analysis. A large portion of the errors did not concern the RMRS as such, but simply the part-of-speech tags, encoded in the relation names. If part-of-speech errors are ignored, the number of correct RMRSs increases from 41% to 68%. The results of validation without part-of-speech correction, calculated on a third sample of 100 sentences, are given in Table 1 (bottom).

Significant structural errors arise primarily in the context of modification. This is due to the TIGER annotation scheme. For example, certain adjunct clauses are embedded in main clauses as `mo` dependents, yet the embedding conjunction is, again, annotated as a modifier of the embedded clause. This leads to erroneous analyses. Refinement of the rules could considerably improve accuracy, but distinguishing these cases from ordinary modification is not always possible, due to missing category information.

While modifiers turned out challenging in the mapping from dependencies to semantics, we did not observe many errors in the treatment of arguments: the rules that map dependents to semantic `arg` predicates yield a very precise argument structure.

5 Conclusion

We presented a method for semantics construction that converts dependency structures to RMRSs as they are output by HPSG grammars. By applying this method to the TIGER Dependency Bank, we construct an RMRS Bank that allows cross-framework parser evaluation for German. Our method for RMRS construction can be transposed to dependency banks for other languages, such as the PARC 700 Dependency Bank for English (King et al., 2003). The choice of RMRS also ensures that the semantic bank can be used for comparative evaluation of HPSG grammars with low-level parsers that output partial semantics in terms of RMRS, such as the RASP parser of (Carroll and Briscoe, 2002).

While the formalism of (R)MRS has its origins in HPSG, we have shown that RMRS semantics construction can be carried over to dependency-based frameworks like LFG. In future research, we will investigate how the semantic algebra of (Copestake et al.,

[10] This evaluation did not perform correction of part-of-speech tags (cf. below, error analysis).

2001) compares to Glue Semantics (Dalrymple, 1999). Our construction rules may in fact be modified and extended to yield semantics construction along the lines of Glue Semantics, with hooks as resources and RELS, HCONS and ING sets as meaning language. In this scenario, the composition rules would consume the hook of the semantic argument, so that resource-sensitivity is assured. Scope ambiguities would not result in alternative derivations, since RMRS makes use of scope underspecification in the meaning language.

Related work in (Dyvik et al., 2005) presents MRS construction from LFG grammars in a *correspondence* architecture, where semantics is defined as a projection in individual syntactic rules. Our architecture follows a *description-by-analysis* (DBA) approach, where semantics construction applies to fully resolved syntactic structures. This architecture is especially suited for the present task of treebank creation, where grammars for a given language may not have full coverage. Also, in a DBA architecture, incomplete rule sets can still yield partially annotated, i.e., unconnected semantic structures. Likewise, this construction method can deal with partially analysed syntactic input.

Finally, our method can be extended to a full parsing architecture with deep semantic output, where care should be taken to preserve structural or categorial information that we identified as crucial for the purpose of principle-driven semantics construction.

Acknowledgements

The research reported here was conducted in cooperation with the TIGER project and has partially been supported by the project QUETAL (DFKI), funded by the German Ministry for Education and Research, grant no. 01 IW C02. Special thanks go to Dan Flickinger and Berthold Crysmann for advice on theoretical and grammar-specific issues. We also thank Martin Forst, who provided us with the TIGER DB dependency structures, Berthold Crysmann for providing HPSG lexical resources, and Ulrich Schäfer for XSLT-scripts used for visualisation.

References

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER Treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol, Bulgaria.

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

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

A. Copestake, D. Flickinger, I. Sag, and C. Pollard. 2005. Minimal Recursion Semantics. to appear.

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

R. Crouch. 2005. Packed Rewriting for Mapping Semantics to KR. In *Proceedings of the Sixth International Workshop on Computational Semantics, IWCS-05*, Tilburg, The Netherlands.

B. Crysmann. 2003. On the efficient implementation of German verb placement in HPSG. In *Proceedings of RANLP 2004*, Bulgaria.

M. Dalrymple, editor. 1999. *Semantics and Syntax in Lexical Functional Grammar: The Resource Logic Approach*. MIT Press.

H. Dyvik, V. Rosén, and P. Meurer. 2005. LFG, Minimal Recursion Semantics and Translation. In *Proceedings of the LFG 2005 Conference*, Bergen, Norway. to appear.

M. Forst, N. Bertomeu, B. Crysmann, F. Fouvry, S. Hansen-Schirra, and V. Kordoni. 2004. Towards a Dependency-Based Gold Standard for German Parsers: The Tiger Dependency Bank. In S. Hansen-Schirra, S. Oepen, and H. Uszkoreit, editors, *Proceedings of LINC 2004*, Geneva, Switzerland.

M. Forst. 2003. Treebank Conversion – Establishing a testsuite for a broad-coverage LFG from the TIGER treebank. In *Proceedings of LINC'03*, Budapest, Hungary.

A. Frank and K. Erk. 2004. Towards an LFG Syntax–Semantics Interface for Frame Semantics Annotation. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, LNCS, Vol. 2945. Springer, Heidelberg.

T.H. King, R. Crouch, S. Riezler, M. Dalrymple, and R. Kaplan. 2003. The PARC 700 Dependency Bank. In *Proceedings of LINC 2003*, Budapest, Hungary.