Chapter 5

The TiGer Dependency Bank — A Dependency-Based Gold Standard for German Parsers

In this chapter, we document how the TIGER-derived packed f-structure representations were disambiguated and hand-checked for the creation of a gold standard for German parsers, the TiGer Dependency Bank, comparable in granularity and representation format to the f-structures produced by the German *ParGram* LFG and hence usable for the evaluation of our system. We especially address the differences between the TIGER Treebank graphs and the structures annotated in the TiGer Dependency Bank.

5.1 A gold standard for (hand-crafted) German parsers

As noted at the beginning of Chapter 4, syntactically annotated data are indispensable for an informative (and potentially comparative) evaluation of parsers. In other words, in order to determine the quality of a parser's output, we need a gold standard for German parsers, suited both for treebank-induced parsers/grammars and for hand-crafted grammars. Given that there are data collections like the TIGER Treebank, one may be tempted to use these. However, the graphs of the TIGER Treebank themselves are difficult, if not impossible, to use as a gold standard for German parsers that were not induced from this same treebank for a number of reasons: The constituency annotation in the TIGER graphs has the advantage of being fairly theory-neutral, but (i) since it includes discontinuous constituents and secondary edges, it cannot be mimicked by any of the hand-crafted German parsers that we are aware of. Besides, (ii) the tokenization (and lemmatization) of certain multi-word ex-

pressions, compounds etc. differs from the analyses most hand-crafted parsers obtain. The functional annotation in the TIGER graphs is more suitable for the evaluation and comparison of parsers across theoretical frameworks, but (iii) it is intimately tied to the constituency annotation, including edge labels such as ADC[1] (multi-token adjective component), which only exist due to the lemmatization decisions mentioned above, and (iv) it does not encode all information and distinctions that deep parsers are supposed to obtain.

Similar problems arise with other syntactically annotated corpora of German text and speech, such as the NeGra Corpus (Skut et al. 1998, Brants et al. 1999), the Verbmobil Corpus (Wahlster 2000) and the Tübingen Treebank of Written German (Telljohann et al. 2003), since they all encode constituency and dependency information in one structure, the latter being biased by the former. Grammar developers, however, are interested in pure dependency representations, which allow for a much more meaningful evaluation than the bracketing of constituents, and have therefore clearly been moving away from treebanks to dependency banks (Carroll et al. 1999, 2003).

Therefore, it was decided in the TIGER Project to establish a purely dependency-based gold standard for German parsers for a part of the TIGER Corpus. The size of this gold standard was determined to be of 2,000 sentences, and the corpus section from sentence # 8,001 to sentence # 10,000 was randomly selected as to be annotated with the corresponding dependency annotations. This gold standard is called the TiGer Dependency Bank (henceforth TiGer DB), and since 132 sentences in the corpus section mentioned either consist of just one word or of a sequence of syntactically unconnected words, it comprises 1,868 structures. The TiGer DB is annotated with so-called dependency triples, i.e. a functor representing a grammatical relation or feature and two arguments representing the head and the value of this feature respectively. The format of these dependency triples is the same as in the PARC 700 Dependency Bank (King et al. 2003), which makes it possible to use the tools for displaying and pruning structures that are available together with this English dependency bank, which was also constructed for the purpose of parser evaluation. The grammatical relations encoded in the TiGer DB are to a fair extent identical to the edge labels used in the TIGER Treebank; in order to make it more suitable as a basis for the evaluation of deep German parsers, additional distinctions have to be made in the set of grammatical relations, however, which leads to an enlarged set of features compared to the TIGER Treebank.

[1]Upper case labels are used for the functional annotation encoded in the TIGER Treebank and lower case labels for the dependencies encoded in the TiGer DB.

5.2 Constructing the TiGer DB

For the construction of the TiGer DB, an approach consisting in a combination of automatic and manual techniques was chosen. The idea was to achieve the most accurate and consistent results in a reasonable amount of time. The basic process is as follows:

1. Convert each TIGER Treebank graph into an f-structure chart (packed representation of one or several f-structures) according to the method presented in Chapter 4.
2. Match the resulting f-structure chart against the output of the German *ParGram* LFG (again according to the method laid out in Chapter 4) and bank the compatible reading(s).
3. For all sentences for which there are either several or no compatible readings, select the correct/best analysis manually.
4. Fully automatically convert the selected f-structure into dependency triples.
5. Manually check/correct each structure using the pretty-printing and validation tools that are distributed with the PARC 700 Dependency Bank.

5.2.1 Automatic derivation of f-structure charts from the TIGER Treebank

The conversion of TIGER graphs into f-structure charts is described in Chapter 4. It takes the TIGER graphs encoded in TIGER XML as input and produces f-structure charts. The ambiguity in the mapping from TIGER graphs to f-structure charts is because of missing information in the TIGER Treebank, such as information concerning the decomposition of compounds, the argument vs. adjunct status of phrases labeled as `MO`s (modifiers), etc. In the conversion process it can be dealt with by means of optional rules, but in order to be used as a gold standard, the resulting output has to be disambiguated, of course. How this can be done is discussed in Subsections 5.2.2 and 5.2.3.

Apart from changes due to the shift from one representation to another, we decided to perform some changes to the analyses chosen by the TIGER Treebank annotators as well. These latter changes are motivated by the fact that the treatment given to these phenomena by all German parsers that we are aware of differs from the analysis in the TIGER Treebank in a systematic way. One of these changes concerns PPs that are extracted from NPs, such as *statt dessen* in (5.1).

(5.1) Statt dessen gestand ihnen die Regierung eine Entschädigung zu:
Instead this-GEN conceded them the government an indemnity to.
'Instead (of this), the government conceded them an indemnity.'[2]

In the TIGER Treebank, this PP is attached as an `MNR` (noun modifier) to the NP *eine Entschädigung*. Current hand-crafted parsers for German, however, would attach this PP to the verb, since the attempt to attach it to the NP would result in a massive increase in ambiguity. For a gold standard for German parsers, we consider it reasonable to encode the latter attachment rather than one that no parser would be able to achieve and which, moreover, is semantically debatable.

5.2.2 Automatic disambiguation of TIGER-derived f-structure charts

As a first step towards disambiguating the f-structure charts resulting from the fully automatic treebank conversion, these are matched against the output of the German *ParGram* LFG, as described in Section 4.7 of the last chapter, and the compatible reading(s) are saved. Of course, this matching can only be performed for sentences that are assigned a full parse by the German *ParGram* LFG, and although the information both in the TIGER graphs and in the LFG parses is relatively detailed, it can be impossible to fully disambiguate. Typical remaining ambiguities are due to the decomposition of compounds and to person and number ambiguities of possessive determiners and pronouns, these pieces of information not being included in the TIGER Treebank.

Moreover, it has to be kept in mind that the matching against the LFG output does not always retain the correct analysis, although most ambiguities can be resolved correctly in this way. This is particularly true for the `mo` (modifier) vs. `op` (prepositional object) distinction; an incomplete lexicon entry in the LFG can lead to the selection of the `mo` reading, even if the `op` reading is more adequate.

Nevertheless, the automatic matching of the TIGER-derived f-structure charts against the output of the German *ParGram* LFG is extremely useful. Not only does it help to eliminate, or at least reduce, the ambiguity in the representations, but it also helps to increase consistency in the gold standard, since every time a match between the TIGER-derived f-structure chart and the grammar output is expected but cannot be achieved, the human annotators can pay special attention to the phenomenon that caused the match to fail.

[2] s14841

5.2.3 Manual disambiguation of TIGER-derived f-structure charts

All ambiguous TIGER-derived f-structure charts that cannot be fully disambiguated in the previous step have to be disambiguated manually. This is performed by visualizing the structures in the grammar development tool XLE (Maxwell & Kaplan 1993, Crouch et al. 2006). It displays the packed representation of all f-structures encoded, the currently selected f-structure and an additional window, where the alternatives with the information differing among them are visualized. This allows human annotators to choose and save the correct reading. When none of the readings can be considered correct, the best analysis is selected and the annotator puts the sentence number on a list of structures to be reconsidered in the validation step.

For mildly ambiguous structures, this manual disambiguation step can be carried out relatively comfortably with the help of XLE. However, when disambiguating highly ambiguous structures, the annotators found it extremely hard to select exactly the correct f-structure, since XLE displays the discriminants between readings in a fashion that is difficult to read in the case of highly ambiguous structures and it does not fix choices that the annotator has made. In conclusion, XLE, which is *not* and has never been conceived as a treebanking tool, can reasonably be used for treebanking when the structures involved are mildly ambiguous; for handling more complex representations, however, it proved to be cumbersome to use. For future annotation efforts similar to the construction of the TiGer DB, the tools developed in the TREPIL Project of the University of Bergen (Rosén et al. 2005), will probably provide an interesting alternative.

5.2.4 Conversion into dependency triples and validation

The conversion from f-structures to dependency triples is fully automatic and unambiguous. It is carried out in basically the same way as it was done for the PARC 700 Dependency Bank (King et al. 2003). It mainly involves a certain amount of "flattening", i.e. articulate f-structures without a PRED have to be restructured, but this can be done without any loss of information. In addition to the flattening, a certain amount of renaming and reorganizing has to be carried out in order to make the structures meet the annotation principles outlined in Subsection 5.3.

In a final (and very important) step, each TiGer DB structure is manually evaluated by an expert. If the structure is not correct, changes are made in the text-based representation of the structure. For this step, it is essential to use the pretty-printing and validation tools that are distributed with the PARC 700 Dependency Bank, as the visualization they facilitate speeds up the validation process considerably. Nevertheless, the manual effort required in this final step

was more than we had expected initially. To give a rough estimate, it probably involved more than six person months.

5.3 Grammatical relations and features encoded in the TiGer DB

The choice of the format and the dependencies encoded in the TiGer DB is crucial for its possible uses. Therefore the contents of the TiGer DB structures themselves are discussed in this subsection. First we discuss indices, reentrancies and lemmatization. We then present the grammatical relations we have decided to encode in the TiGer DB and finally the atomic features chosen.

5.3.1 Indices, Reentrancies and Lemmatization

Just as in the PARC 700 Dependency Bank, all predicates in a given TiGer DB structure are assigned a unique index. For displaying reasons, the matrix predicate is always assigned the index 0. All other predicates are assigned the index corresponding to the ID of the terminal node in the TIGER Treebank that it relates to. Predicates which do not clearly relate to a terminal node in the TIGER Treebank are given a 'new' arbitrary index. (This is the case for the compound non-head *privat∼1001* in Figure 5.2 on p. 46, for example.)

The use of indices has a number of advantages: First, they help to distinguish two instances of the same word. Second, they permit the expression of reentrant structures, i.e. structures in which a single item is related to more than one predicate. This occurs with controlled infinitives and with predicative constructions.

Consider the sentence in (5.2) as well as its TIGER graph representation (in TIGER XML) and its representation as dependency triples, which are shown in Figures 5.1 and 5.2 (p. 46) respectively.

(5.2) Privatmuseum muß weichen
Private museum must leave
'Private museum must leave'[3]

The matrix predicate of the sentence is the verb *müssen*; this verb is thus assigned the index 0. All other predicates are assigned the indices corresponding to the terminal nodes they relate to, which are 1 for *(Privat)Museum* and 3 for *weichen*. The 'new' predicate *privat*, whose existence is due to the decomposition of compounds in the TiGer DB, is assigned a new unique index calculated on the basis of the index of its head and the position of the compound non-head

[3] s8597

```
<s id="s8595">
 <graph root="s8595_500">
  <terminals>
   <t id="s8595_1"
     word="Privatmuseum"
     pos="NN" morph="Nom.Sg.Neut"/>
   <t id="s8595_2" word="muß"
     pos="VMFIN" morph="3.Sg.Pres.Ind"/>
   <t id="s8595_3" word="weichen"
     pos="VVINF" morph="--" />
  </terminals>
  <nonterminals>
   <nt id="s8595_500" cat="S">
    <edge label="SB" idref="s8595_1"/>
    <edge label="HD" idref="s8595_2"/>
    <edge label="OC" idref="s8595_3"/>
   </nt>
  </nonterminals>
 </graph>
</s>
```

Figure 5.1: TIGER XML representation of (5.2)

within the compound, which turns out to be 1001 in the present example. The fact that the subject of the embedded verb *weichen* shares its structure with the subject of the top verb *müssen* is expressed by the two triples `sb(müssen~0, Museum~1)` and `sb(weichen~3, Museum~1)`.

Concerning the grammatical relations encoded, the dependency triple representation shows both similarities and differences to the TIGER XML graph representation. Just as the latter, it annotates *(Privat)Museum* as the `sb` (subject) of *müssen*, but in contrast to the graph, it also annotates it as the `sb` of *weichen*. In addition, the `OC` (clausal object) edge label from the graph is reinterpreted as an `oc_inf`, since the related phrase is an infinite clausal object.

The above example also demonstrates the lemmatization applied in the TiGer DB. Verb forms are lemmatized to the infinitive, nominal forms to the nominative singular etc. Compounds are split up into their components, of which the head is used in the predicate name and the others are `mod` dependents of this head. In order to keep track of the original compound form, the lemma of the compound is encoded as the value of the feature `cmpd_lemma`, as can be seen in Figure 5.2 (p. 46).

```
case(Museum~1, nom),
cmpd_lemma(Museum~1, Privatmuseum),
gend(Museum~1, neut),
mod(Museum~1, privat~1001),
mood(müssen~0, indicative),
num(Museum~1, sg),
oc_inf(müssen~0, weichen~3),
pers(Museum~1, 3),
sb(müssen~0, Museum~1),
sb(weichen~3, Museum~1),
tense(müssen~0, pres)
tiger_id(müssen~0, 2)
```

Figure 5.2: Dependency triple representation of (5.2)

5.3.2 Grammatical relations

The most difficult decisions in creating the TiGer DB involve choosing the grammatical relations to be encoded. Which dependencies are needed in the final application differs from framework to framework, and the names they are given vary from grammar to grammar. As a guideline, it has been decided to stick to the functional annotation in the TIGER Treebank, i.e. the edge labels in the TIGER graphs. Additional distinctions were introduced where the TIGER Treebank annotations seemed not to make all the distinctions current deep parsers of German make. The most striking example of a TIGER Treebank edge label which is treated in a number of different ways by German parsers is `MO`, which can be a truly optional modifier (still labeled as `mo` in the TiGer DB), but also a predicative argument (labeled as `pd` in the TiGer DB) or a (more or less) obligatory directional or locative argument (labeled as `op_dir` and `op_loc` respectively).

Grammatical relations defined like in the TIGER Treebank

The grammatical relations that are encoded identically in the TIGER Treebank (or in a former version of it) and in the TiGer DB are:

- `ams` – measure phrases that modify adjectives
- `cj` – conjunct of a coordination
- `da` – objects in the dative
- `oa` – direct objects in the accusative
- `oa2` – secondary objects in the accusative

- og – objects in the genitive
- op – prepositional objects pseudo-genitives
- rc – relative clauses
- sbp – logical subjects of verbs in the passive
- vo – vocatives

Subjects

Although all SBs (subjects) of the relevant TIGER Treebank graphs appear as sbs in the corresponding TiGer DB annotations, there is one difference with respect to subjects: The TiGer DB encodes more subjects than the TIGER Treebank. In addition to subjects of inflected verb forms, these are the following:

- **sbs within oc_infs, infinitival sbs and mos:** oc_infs (infinite clausal objects, see page 51), as well as infinitival sbs (subjects) and mos (modifiers, see page 50) *always* contain a sb. In the case of oc_infs of raising verbs, this sb is indicated by coindexation, as in Figure 5.2, corresponding to (5.2), where the sb of the verb *weichen* is coindexed with the sb of the modal verb *müssen*, and Figure 5.3 (p. 48),[4] corresponding to (5.3), where the sb of the verb *festnageln* is coindexed with the oa (accusative object) of the verb *lassen*. In the case of oc_infs of equi verbs and of infinitival sbs and mos, the sb is filled by a null pronoun, as in Figure 5.3, where the sb of the verb *lassen* is a null pronoun, and Figure 5.4 (p. 49), corresponding to (5.4), where this is the case for the sb of the verb *zurückhalten*.

(5.3) . . . , ohne sich auf deren Umfang festnageln zu lassen.
. . . without himself on these-GEN amount nail down to let.
'. . . without letting himself be nailed down to the amount of these.'[5]

(5.4) . . . hat . . . vorgeworfen, wichtige Informationen über Kriegsverbrechen in Bosnien zurückzuhalten.
. . . has . . . accused important informations about war crimes in Bosnia to withhold.
'. . . has accused . . . of withholding important information about war crimes in Bosnia.'[6]

[4] For better readability, we show all following sample TiGer DB structures as they are displayed by the pretty-printing tool that comes with the PARC 700 DB.
[5] s9966
[6] s9034

```
| mo  | pred ‘ohne’
      | obj  | pred ‘lassen’
             | oa  [16]| pred ‘pro’
                       | case acc
                       | num sg
                       | pers 3
                       | pron_type refl
             | oc_inf  | pred ‘fest#nageln’
                       | pass_asp modal
                       | op  | pred ‘auf’
                             | obj  | pred ‘Umfang’
                                    | case acc
                                    | gend masc
                                    | num sg
                                    | gl  | pred ‘pro’
                                          | case gen
                                          | gend fem
                                          | num sg
                                          | pron_type demon
                       | sb  [16: pro]
             | sb  | pred ‘pro’
                   | pron_type null
```

Figure 5.3: Dependency triple representation of (5.3)

- `sb`**s within** `pd`**s**: Whether a predicative argument (`pd`) is subject-controlled or object-controlled is *always* indicated by a coindexed `sb`. This is illustrated in Figure 5.5 (p. 50), corresponding to (5.5), where the dependency triple `sb(mitverantwortlich~10, Regierung~9)` encodes that the `pd` of the verb *machen* is object-controlled.

(5.5) Der DIHT macht die Regierung für die eingetrübte Stimmung mitverantwortlich.
The DIHT makes the government for the tarnished vibes co-responsible.
‘The DIHT holds the government for co-responsible for the tarnished vibes.’[7]

[7] s9992

```
| pred ‘vor#werfen’
| mood ind
| perf +
| tense pres
| oc_inf  | pred ‘zurück#halten’
          | oa  | pred ‘Information’
                | case acc
                | gend fem
                | num pl
                | mo  | pred ‘wichtig’
                      | degree pos
                | op  | pred ‘über’
                      | obj  | pred ‘Verbrechen’
                             | case acc
                             | cmpd_lemma Kriegsverbrechen
                             | gend neut
                             | num pl
                             | mod  | pred ‘Kriegs’
                             | mo  | pred ‘in’
                                   | obj  | pred ‘Bosnien’
                                          | case dat
                                          | gend neut
                                          | num sg
          | sb  | pred ‘pro’
                | pron_type null
```

Figure 5.4: Dependency triple representation of (5.4)

Grammatical relations whose definition diverges from the one in the TIGER Treebank

Grammatical relations that can also be found in the TIGER Treebank, but whose definition diverges from the one there, are:

- `app` – close appositions, opposed to wide appositions in the TIGER Treebank; the latter are shifted to `mo`.
- `cc` – comparative (and equative) complements; in contrast to `CC` in the TIGER Treebank, these no longer comprise *wie*-PPs that are not triggered by an equative context; not being subcategorized, these are treated as `mos` in the TiGer DB.
- `gl` – genitive attribute on the left of its head noun *or* possessive determiner.

```
| pred ‘machen’
| oa  [9]| pred ‘Regierung’
         | det  | pred ‘die’
| pd  | pred ‘mitverantwortlich’
      | op  | pred ‘für’
            | obj  | pred ‘Stimmung’
                   | det  | pred ‘die’
                   | mo  | pred ‘ein#trüben’
      | sb  [9: Regierung]
| sb  | pred ‘DIHT’
      | det  | pred ‘die’
```

Figure 5.5: Predicate-argument triples of (5.5)

- `mo` – optional modifiers; in contrast to `MO` in the TIGER Treebank, `mo` in the TiGer DB no longer comprises (more or less) obligatory directional, local and modal arguments; the definition is enlarged with respect to the TIGER Treebank in that it now includes `APP`s (wide appositions) and `CC`s (comparative complements) that are not triggered by a comparative or equative context.

- `gr` – genitive attribute on the right of its head noun; in contrast to `GR` in former versions of the TIGER Treebank, `gr` also comprises PPs introduced by *von* which are annotated as `PG`s in the treebank.

- `pd` – all predicative arguments, not only those of the copular verbs *bleiben*, *sein* and *werden*.

- `rs` – reported speech, in sentences like (5.6), where the `RS` clause is not regularly subcategorized for by the matrix verb; note that all other `RS` constructions of the TIGER Treebank are reinterpreted as `oc_fin`s (finite clausal objects, see 51).

(5.6) Technisch sei dies machbar, widersprach Starzacher den Skeptikern
Technically is this feasible, contradicted Starzacher the scepticists
in der Verwaltung.
in the administration.
‘Starzacher contradicted the skeptics in the administration, saying that, technically, this was feasible.’

'New' grammatical relations

Finally, there are a number of 'new' grammatical relations in the TiGer DB, which arise from the more fine-grained distinctions that are made in the dependency bank with respect to the TIGER Treebank edge labels.

- `app_cl` – appositive clauses, occurring with *es* and pronominal adverbs
- `det` – articles, demonstrative and interrogative determiners
- `measured` – measured entity in constructions such as (5.7)

(5.7) zwei Flaschen Wein
two bottles wine
'two bottles of wine'

- `mod` – non-head components of compounds (see, e.g., Figure 5.4 on p. 49)
- `name_mod` – non-head in complex name
- `number` – numbers in specifier position
- `obj` – argument of a preposition or a subordinating conjunction
- `oc_fin` – finite clausal objects (*dass/ob*-clauses, indirect wh-questions)
- `oc_inf` – infinite clausal objects
- `op_dir` – directional oblique arguments
- `op_loc` – local oblique arguments
- `op_manner` – modal oblique arguments
- `pred_restr` – sublexical dependency from the *pro* predicate of a nominalized adjective or participle to the actual lemma of the adjective or participle
- `quant` – quantifying specifiers
- `topic_disloc` – left dislocated phrases

In general, determining these grammatical relations is relatively straightforward. There are exceptions to this rule, however, such as the distinction between `mos` (modifiers) and the different `ops` (prepositional arguments), as well as certain constructions where a given PP could be analyzed either as a `mo` or a `pd` (predicative argument).

5.3.3 Atomic features

The atomic features included in the TiGer DB correspond mostly to the morphological information encoded in the TIGER Treebank, but also to information from the part-of-speech tags. Furthermore, some of them encode the form of words that do not introduce a predicate themselvers. Generally the atomic features further specify the predicates that relate to the terminal nodes in the TIGER Treebank where the information is encoded. An exception to this rule is the person/number agreement information given for finite verb forms, which ends up in the features `num` and `pers` of the subject of the verb under consideration, as well as agreement information provided by determiners and inflected adjectives, which is attached to their head noun. The purpose of this is to avoid the doubling of information.

It was decided to encode the following atomic features in the TiGer DB:

- `case` with the values `acc` (accusative), `dat` (dative), `gen` (genitive) and `nom` (nominative)
- `circ_form` with the values `an`, `aus` and `willen`, provided by the final part of circumpositions
- `cmpd_lemma` (see Subsection 5.3.1)
- `comp_form` (complementizer form) provided by *dass* or *ob*)
- `coord_form` (form of coordinating conjunction) provided by *aber*, *oder*, *und* etc.
- `degree` with the values `pos` (positive), `comp` (comparative) and `sup` (superlative)
- `det_type` (type of determiner) with the values `def` (definite), `demon` (demonstrative), `indef` (indefinite) and `int` (interrogative)
- `fut` (future) with the value + (otherwise unspecified)
- `gend` (gender) with the values `fem` (feminine), `masc` (masculine) and `neut` (neuter)
- `mood` with the values `imp` (imperative), `ind` (indicative) and `subj` (subjunctive)
- `num` (number) with the values `sg` (singular) and `pl` (plural)
- `pass_asp` (dynamic vs. stative passive) with values `dynamic` and `stative`
- `perf` (perfect) with the value + (otherwise unspecified)

- `pers` (person) with values `1`, `2` and `3`
- `precoord_form` (first part of composite coordinating conjunction) provided by *entweder*, *sowohl* etc.
- `pron_form` (form of indefinite pronoun) with values like `etwas`, `jemand`, `man`, `nichts` etc.
- `pron_type` (type of pronoun) with the values `demon` (demonstrative), `int` (interrogative), `null` (non-overt), `pers` (personal), `quant` (quantifying), `recip` (reciprocal), `refl` (reflexive) and `rel` (relative)
- `tense` with the values `pres` (present) and `past`

5.4 Uses of the TiGer DB

The TiGer DB is designed as a gold standard for the dependency-based evaluation of German parsing systems. Since it uses the fairly theory-independent dependency structures (even though they are labelled), we expect them to be of use for a number of linguistic theories and, hence, to allow cross-framework comparisons. Concerning the concrete possibilities of matching parser output against the TiGer DB dependency triples, we have considered this task both for an LFG and for an HPSG parser.

Given the resemblance between the TiGer DB representations and f-structures, the mapping from LFG f-structures to dependency triples is straight-forward. It involves some renaming and reorganizing of the structures, but this is basically the same as in the final step of the construction of the TiGer DB (see Subsection 5.2.4).

The mapping from HPSG feature structures to the TiGer DB is less trivial, since the representations differ more. Nevertheless, the TiGer DB was constructed in close collaboration with the HPSG developers at Saarland University, so that it should, at least in principle, be a useful resource for the HPSG community as well, although a direct mapping from the (R)MRSs produced by the German HPSG developed in Saarbrücken seems to be difficult. Spreyer & Frank (2005a,b) therefore investigated ways of converting the TiGer DB structures consisting of dependency triples into RMRSs by applying a further transfer step to them.

Due to its relatively limited size, the TiGer DB has so far only been used for evaluation purposes and, at least in as far as our grammar development activities are concerned, it will continue to be considered evaluation data and thus not be inspected during grammar development.

In order to enable us to use TIGER-derived dependency annotations for corpus sections other than sentence # 8,001 to sentence # 10,000 in regression tests during grammar development, a fully automatic conversion of TIGER

Treebank graphs into dependency triples has recently been developed (Kountz 2006). The grammatical relations used in these new representations correspond directly to TIGER Treebank edge labels, so that the representations are more coarse-grained than the TiGer DB structures. However, they are available for almost the whole TIGER Corpus, and thanks to a recent extension of the XLE output representations, we will be able to map the f-structures produced by the German *ParGram* LFG onto this new type of dependency structure. For the first time, we will then be able to closely observe the effects of grammar modifications on the accuracy of the grammar's analyses during grammar development. Final evaluations will be carried out both on the more fine-grained TiGer DB and on the new dependency structures.

5.5 Summary

The TiGer Dependency Bank is a dependency bank containing both grammatical relations between predicates and arguments and a number of other grammatical features. In this, it is closely related to the PARC 700 Dependency Bank (King et al. 2003). It has been produced semi-automatically on the basis of the TIGER Treebank annotations, partly cross-validated by means of the German *ParGram* LFG and finally validated by human annotators. The automation and cross-validation allow for a rapid construction of the TiGer Dependency Bank compared to the creation of such a resource from scratch. Nevertheless, the manual effort involved in resolving ambiguities introduced during lemmatization and the reinterpretation of ambiguous TIGER edge labels was considerable.

The TiGer DB is intended to be used for the evaluation of German parsers. As the grammatical relations encoded in it are close to the ones in the TIGER Treebank and the related NeGra Treebank, we hope that, apart from the German *ParGram* LFG, all kinds of parsing systems for German, both hand-crafted ones and systems that were induced from the treebanks mentioned, will be evaluated on it. In order to facilitate this, this chapter has given a relatively detailed account of how the TiGer DB structures relate to the corresponding TIGER Treebank graphs.