

The TiGer Dependency Bank in Prolog format

Tomas By

Centro de Linguística da Universidade Nova de Lisboa
Avenida de Berna 26-C, 1069-61 Lisboa, Portugal
tomas.by@fcsh.unl.pt

Abstract

The TiGer dependency bank, like the PARC 700 dependency bank, has lemmatised tokens and no representation of word order, meaning that multiple occurrences of the same word type in one sentence are ambiguous. The number of such ambiguous word tokens is around 9%, compared to 15% in PARC 700. This paper reports a conversion of the TiGer dependency bank to a Prolog representation that represents the word order explicitly, uses the surface forms of the words rather than base forms, makes a clear distinction between words and empty nodes, and stores the token attributes in one place rather than spread out across the file. Together with the Prolog format data, graphical representations of the dependency trees are provided in PDF. Hopefully this work will make it easier to use the TiGer dependency bank data for parser evaluation of German.

Keywords: dependency grammar, parser evaluation

1 Introduction

The TiGer dependency bank (Forst *et al.*, 2004) is a conversion of 1865 sentences¹ from the TiGer Treebank to a dependency format based on the one used in the PARC 700 dependency bank (King *et al.*, 2003). This format is intended for parser evaluation using the method described in Carroll *et al.* (1998), namely converting the parser output to a set of dependency relations between strings representing the base forms of some of the words in the sentence, and comparing these to the supposedly correct ones in the dependency bank. As pointed out in By (2007), this fails to take into account that a sentence may contain more than one instance of the same word type, which may lead to comparisons against the wrong dependency relations. In the PARC 700 dependency bank, around 15% of the tokens are ambiguous in this way (By, 2007, pp. 275–7), and in the TiGer dependency bank it is about 9%. Other problems with the PARC 700 format, inherited by the TiGer dependency bank and not shared by the Prolog format described here, include a spurious distinction between full tokens and attribute tokens, the absence of a distinction between words and empty nodes in the dependency tree, and a

¹There are 1867 files, but two pairs are duplicates, cf. note 4 on p. 120. The version used here is dated August 2007 (<http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERCorpus/download/TiGerDB-Aug07.zip>).

unnecessarily distributed representation of the token attributes. After sections on problems with the PARC 700 format, differences between the TiGer and PARC 700 dependency banks, differences between TiGer depbank and treebank, and other problems in the TiGer dependency bank, there is one section describing the conversion from the TiGer format into Prolog, and one section on this Prolog format. Finally there is a summary.

2 Problems with the PARC 700 format

In the representation suggested in Carroll *et al.* (1998), which was the inspiration for the PARC 700 Dependency Bank (Crouch *et al.*, 2002, p. 67), the dependency relations are relations between strings. As Crouch *et al.* (2002, p. 71) and King *et al.* (2003, p. 4) point out, multiple occurrences of the same word in a sentence can then not be distinguished. The proposed solution, in the PARC 700 and adopted by the TiGer dependency bank (Forst *et al.*, 2004, p. 33), was to add a numerical index to each of the words that participate in dependency relations. But this device only serves to partition the data internally in the dependency bank, it does not link it to the different occurrences of the word in the sentence. The missing information is the word-order. Figures 1 and 2 (By, 2007, p. 266) illustrate graphically what effect incorrect disambiguation of the word order has on the dependency structure, showing two analyses of the same sentence differing only in that the placement of the word ‘crowded’ has been swapped. The total number of word tokens in the PARC 700 dependency bank that are affected in this way is 15% (By, 2007, pp. 275–7). Table 1 shows similar data for the TiGer dependency bank. For reasons of space, only word (types) that occur ambiguously more than four times² are shown in the table. The total number of ambiguous tokens is 3032 out of 34076, which is 8.9%.

Like the PARC 700, the TiGer dependency bank has symbolic tokens, or ‘empty nodes’ in the tree, that do not correspond to words in the sentence.³ Examples of this are the ‘pro’ nodes in sentence 9433 (figure 3).⁴ In the file format used in these two dependency banks, there is no indication whether a token corresponds to a word or not, so this is a further source of ambiguity.⁵

²i.e. twice in one sentence only, three times in once sentence, or twice in two sentences.

³As pointed out in By (2007, p. 265), this is not traditionally used in dependency grammar Hays (1964); Gaifman (1965); Robinson (1970). Tesnière (1980, pp. 46–7) and Mel’čuk (1988, p. 15) seem to allow them, but only as terminal nodes.

⁴In the TiGer dependency bank, sentence numbers appear both in the file name (e.g. `tiger-db-8001.fdsc`) and in the ‘id’ field (e.g. `TiGerDB_8001`). These numbers do not always agree. Comparison with the XML version of the TiGer Treebank shows that the number in the file name is the correct one in all cases except two, where the ‘id’ field is correct instead. The two cases are ‘s8168/tiger-db-8172.fdsc’ and ‘s9198/tiger-db-9189.fdsc.’ The files ‘tiger-db-8168.fdsc’ and ‘tiger-db-8172.fdsc’ are identical, and the files ‘tiger-db-9189.fdsc’ and ‘tiger-db-9198.fdsc’ are identical except for whitespace. Sentence numbers used in this paper always refer to the file, as the file names are known to be unique.

⁵In the PARC 700 there is one occurrence of ‘pro’ as a real word (By, 2007, p. 278). In the TiGer dependency bank there are three: sentence 8423, ‘pro Jahr;’ 8818, ‘Kilometern pro Stunde;’ 9281, ‘pro Soldat.’

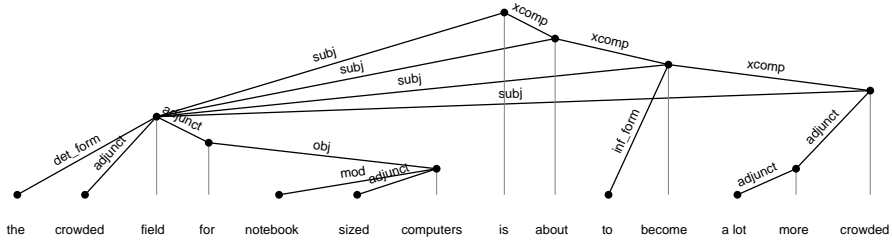


FIGURE 1: PARC #284

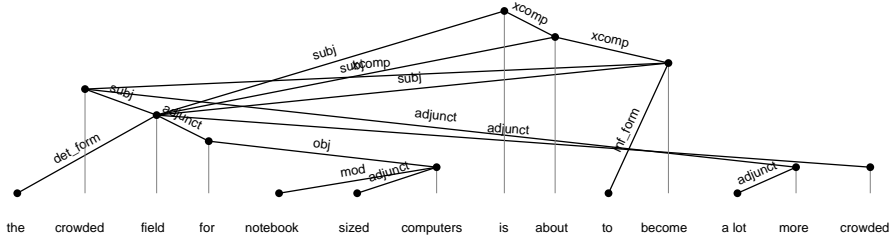


FIGURE 2: PARC #284 with the word 'crowded' incorrectly disambiguated

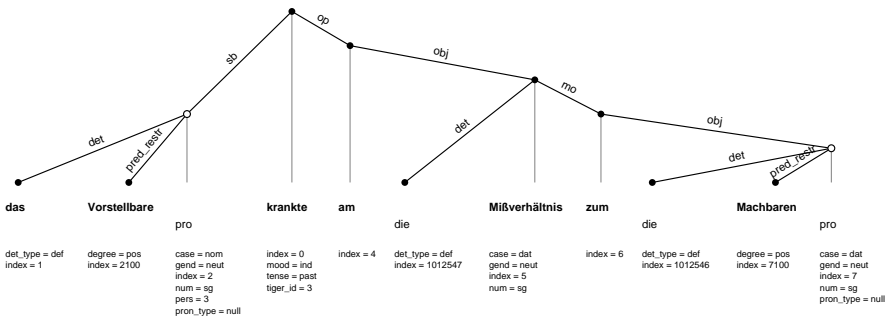


FIGURE 3: TiGer sentence 9433 as a graphical tree

TABLE 1: Tokens that occur multiple times in the same sentence

Token	Number of occurrences (in one sentence)								
	Twice	3 times	4 times	5 times	6 times	7 times	8 times	9 times	
als	13	—	—	—	—	—	—	—	
am	3	—	—	—	—	—	—	—	
an	6	—	2	—	—	—	—	—	
Arbeit	2	1	—	—	—	—	—	—	
auch	6	1	—	—	—	—	—	—	
auf	16	2	—	—	—	—	—	—	
aus	4	1	—	—	—	—	—	—	
bei	7	—	—	—	—	—	—	—	
bin	3	—	—	—	—	—	—	—	
Bundes	3	—	—	—	—	—	—	—	
da	5	—	—	—	—	—	—	—	
das	25	—	1	—	—	—	—	—	
daß	5	1	—	—	—	—	—	—	
dem	19	1	—	—	—	—	—	—	
den	39	3	1	1	—	—	—	—	
der	207	69	22	6	2	1	—	—	
des	25	—	1	—	—	—	—	—	
die	183	33	13	2	—	—	1	1	
du	4	1	—	—	—	—	—	—	
durch	4	—	—	—	—	—	—	—	
ein	7	3	—	—	—	—	—	—	
eine	11	—	—	—	—	—	—	—	
einem	3	1	—	—	—	—	—	—	
einen	3	—	—	—	—	—	—	—	
einer	10	—	—	—	—	—	—	—	
eines	2	1	—	—	—	—	—	—	
er	10	—	—	—	—	—	—	—	
es	7	1	—	—	—	—	—	—	
für	16	—	—	—	—	—	—	—	
haben	3	—	—	—	—	—	—	—	
ich	13	5	—	—	—	—	—	—	
im	12	—	—	—	—	—	—	—	
in	69	16	5	1	—	—	—	—	
ist	4	1	—	—	—	—	—	—	
man	3	1	—	—	—	—	—	—	
Mark	3	—	—	—	—	—	—	—	
mehr	3	—	—	—	—	—	—	—	
Milliarden	5	—	—	—	—	—	—	—	
Millionen	3	1	—	—	—	—	—	—	
minister	3	—	1	—	—	—	—	—	
mit	20	1	—	—	—	—	—	—	
nach	8	1	—	—	—	—	—	—	
Natur	1	1	—	—	—	—	—	—	
nicht	16	1	—	—	—	—	—	—	
nur	5	—	—	—	—	—	—	—	
oder	3	—	—	—	—	—	—	—	
Prozent	4	1	—	—	—	—	—	—	
schutz	1	1	—	—	—	—	—	—	
sein	3	—	—	—	—	—	—	—	
sich	7	1	1	—	—	—	—	—	
Sicherheits	3	—	—	—	—	—	—	—	
sie	7	1	—	—	—	—	—	—	
so	3	—	—	—	—	—	—	—	
SPD	5	—	—	—	—	—	—	—	
steuer	1	1	—	—	—	—	—	—	
um	5	—	—	—	—	—	—	—	
und	85	15	3	—	—	—	—	—	
unter	3	—	—	—	—	—	—	—	
über	7	—	—	—	—	—	—	—	
von	37	5	—	—	—	—	—	—	
vor	4	1	—	—	—	—	—	—	
wie	4	—	—	—	—	—	—	—	
wird	3	—	—	—	—	—	—	—	
zu	34	5	2	—	—	—	—	—	
zur	5	—	—	—	—	—	—	—	

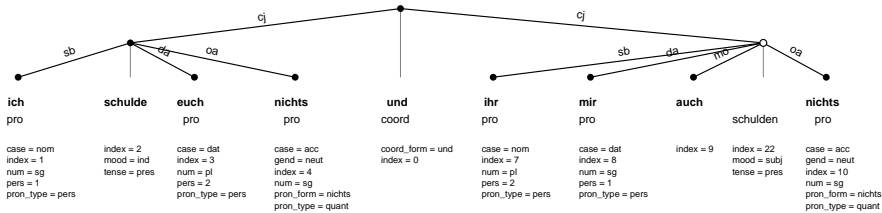


FIGURE 4: Sentence 9572 as a graphical tree

3 Differences between TiGer and PARC 700 depbanks

Naturally, the labels on the dependency links, and the attribute names and values, differ between the PARC 700 and TiGer dependency banks (Forst *et al.*, 2004, pp. 34–5), since the languages are different (English; German), but there are also differences of a more fundamental nature. The perhaps most problematic one, an example of which is shown in figure 4, is that in the TiGer dependency bank, ellipsed words in conjunctions have the exact same representation as the word that has not been ellipsed. In figure 4, the first ‘schulden’ node in the tree corresponds to the word in the sentence, but the second one is an empty node. The decision to show them in this order, and not the opposite, was arbitrary. In the TiGer data there is no difference between them.⁶ There are thirty-one occurrences of this problem in the TiGer dependency bank.⁷ Besides the obvious problem of having two items in the ‘gold standard’ that look like two occurrences of the same word, while there is only one occurrence in the sentence, it seems unlikely that any parser evaluation using this ‘dependency-based’ method will be reliable for sentences with ellipsed verbs. Unless the parser (or a post-parsing data conversion) creates empty nodes in the right places in the tree, those links will not match, and even if the trees are isomorphic, false positives are still possible because of the ambiguity.

Another type of construction that is problematic in the TiGer dependency bank is the representation of ‘fused words’ (Hudson, 2004, p. 100). In figure 3 there are two examples, ‘am’ and ‘zum.’ The former is a contraction of ‘an’ and ‘dem,’ both of which are represented in the TiGer data. The problem, of course, is that there is only one word in the sentence, and that the two nodes are not directly related in the tree. The somewhat similar contractions of negations in English (e.g. wasn’t) are tokenised as two separate units, both in the Penn Treebank and in PARC 700, and they are directly connected in the tree.

The tokenisation of names also differ between the two dependency banks. In PARC 700, they are generally one token, but in TiGer most seem to be several tokens, separated at whitespace.

⁶In the PARC 700 dependency bank, empty nodes like these are labelled ‘null,’ clearly separating them from the real words. (The word ‘null’ does not occur in the data.) Interestingly, in the TiGer dependency bank, which does not use that node label at all, there are two occurrences of ‘null’ as a word (sentences 9454 and 9918).

⁷Sentences 8035, 8065, 8104, 8559, 8678, 8878, 8880 (×2), 9166, 9195, 9280, 9298, 9306, 9381, 9397, 9415, 9439, 9458 (×3), 9546, 9547, 9572, 9638, 9676, 9686, 9692, 9837, 9944 (×2), and 9962.

TABLE 2: Comparison of the tokenisations of TiGer corpora

	Treebank	Depbank	(both)
Identical			26923
Same, except for capitalisation			1220
Morphology			9
Extra punctuation in the Treebank token			45
Space char. in DB, but not TB, token			9
Tokens that correspond directly	28206	28206	28206
Multi-word tokens	2554	262	
Components of multi-word tokens	561	4570	
Tokens with no correspondence	4969	5	
Total number of tokens	36290	33043	

4 Differences between TiGer depbank and TiGer treebank

The TiGer dependency bank was created from the TiGer treebank, which is also available in XML format. A natural application for the Depbank would then be to evaluate a parser designed for the Treebank, in which case differences in tokenisation are an issue. In contrast to the PARC 700 dependency bank (By, 2007, p. 278) the tokenisations in the TiGer case are quite similar, as shown in table 2. The main difference is the ‘multi-word’ tokens, which seem to be mainly compound words in the Treebank that have been split up in the Depbank, and hyphenated words that are individually tokenised in the Treebank but one token in the Depbank. The ‘no correspondence’ tokens are mostly punctuation.

5 Other problems in the TiGer depbank

Table 3 list a number of further problems that were noted when converting the TiGer dependency bank. Most of them seem to be simple spelling mistakes or the like, and in some places there is structure missing. Of this, only the major content words (nouns, verbs) are shown in the table. There are probably other minor words (pronouns, prepositions) missing, but this may not be greatly relevant for parser evaluation. Some word types, like the infinitive marker, are never included in the dependency bank structures (The infinitive markers were added during the conversion to Prolog format).

6 The conversion procedure

The problematic aspect of the conversion is the disambiguation of word types that occur more than once in the same sentence (cf. table 1). For converting the TiGer dependency bank, the same basic approach as is described in By (2007, pp. 268–9) was used. There are two steps. First, the original sentence from the ‘`sentence_form`’ field is tokenised and matched against the dependency bank tokens. This is in effect a search procedure that terminates when a complete tokenisation of the sentence string is found that contains all the tokens that occur

TABLE 3: Problems in the Tiger dependency bank

Sentence(s)	Problem & correction
8076	Changed '1945' to '1968' (as in the <code>sentence_form</code> field, and the Treebank)
8086	Attribute value for 'case,' token 'pro ¹¹ ' (dessen), changed from 'gend' to 'gen'
8104	Duplicate attribute <code>pro³/pron_type = pers & null</code> ; the 'null' removed
8130	Changed token 'auf#auf#nehmen ⁰ ' to 'auf#nehmen ⁰ '
8167	Replaced one token 'die ¹³ ,' which was linked to two different words (Kommunist and Montag), with two tokens 'die ¹³¹ ' and 'die ¹³² '
8282	Changed attribute value '+' to '+' (klettern ⁰ /fut)
8290	Added missing index numbers to two 'cj' clauses (token <code>coord²⁸</code>)
8421	Changed 'werfen' to 'werden' in the <code>sentence_form</code> field
8462, 8829, 9007, 9610, 9709	Duplicate attribute <code>nah/degree = pos & sup</code> (nächsten); the 'sup' removed
8566	Duplicate attribute <code>nah⁹/degree = pos & sup</code> (nächste); the 'sup' removed
8597	Encoding of 'ü' changed from UTF-8 to ISO 8859-1 (as in all other files)
8649	Changed 'Deutsche Bahn AG' to 'Deutschen Bahn AG,' (sentence_form)
8655, 9414	Duplicate attribute <code>weit/degree = pos & comp</code> (weitere); the 'pos' removed
8740	Changed 'Pepräsentant' to 'Repräsentant,' as in the <code>sentence_form</code> field
8798	Duplicate attribute <code>pro²²/pron_type = pers & demon</code> ; the 'demon' removed duplicate attribute <code>pro²²/case = nom & acc</code> ; the 'acc' removed (<code>pro²²</code> is the pronoun 'er' as subject in a subordinate clause)
8799	Changed 'Junden' to 'Juden,' as in the <code>sentence_form</code> field
8924	Changed 'Schmidtbauer' to 'Schmidbauer,' as in <code>sentence_form</code> & Treebank
9217	Changed token 'voraus#voraus#sagen ⁰ ' to 'voraus#sagen ⁰ '
9381	Changed 'infanteritische' to 'infanteristische' (sentence_form)
9383	Replaced one token 'die ⁸ ,' which was linked to two different words (übrig and Bundeswehr), with two tokens 'die ⁸¹ ' and 'die ⁸² '
9572	Duplicate attribute <code>schulden²/mood = ind & subj</code> ; the 'subj' removed
9620	Changed attribute value '+' to '+' (sein ⁰ /fut)
9663	Duplicate attribute <code>Deutschland¹⁶/gend = neut & masc</code> ; the 'masc' removed duplicate attribute <code>NABU¹⁷/gend = neut & masc</code> ; the 'masc' removed
9695	Token 'Absakken' changed to 'Absacken,' as in <code>sentence_form</code> field & Treebank
9769	Token 'eine ¹⁵¹ ' removed. Apparently it corresponds to the first word in 'ein wenig,' but there is also a token 'ein wenig ¹⁵² '
Sentence(s)	Problem (uncorrected)
8077	Directed cycle (rc/pd/obj/sb)
8142	Missing token for the prepositional object '1992'
8151	The (verb) token 'werden ¹⁷ ' has a 'coord_form' attribute
8152	Two tokens (<code>doch⁹</code> & <code>coord⁰</code>) apparently representing the same word (<code>doch</code>)
8278, 9118, 9156	Left conjunct in conjunction missing
8280	Missing token for the prepositional object '1989'
8378, 8615, 9059, 9724, 9855	Directed cycles (sb/pred_restr)
8419, 9424, 9623, 9658	Directed cycles (mo/oc_fin)
8453	No structure at all (but the sentence has two words)
8455, 9281, 9416, 9530, 9870	Incomplete structure, and no token numbered zero
8486, 8493	The personal pronoun 'wir' is marked as third person, not first
8588, 8615	The personal pronoun 'seinen' is marked as plural, not singular
8592	It seems that the tokens 'pro ¹⁰¹ ' and 'inh-refl ¹⁰² ' are the same word (sich)
8712	The 'gr' link from 'Montag ¹⁶ ' to 'FR ¹⁴ ' seems odd
8743	The token 'nicht ²⁰ ' is mis-attached
8752	The modifier '1935' seems mis-attached
8853	Missing token for the prepositional (circumpositional) object '1996'
8981	Missing token for the verb 'umgeleitet'
9281	Incomplete structure
9287	There is both a token 'Streitkraft ²³ ' and a token 'Streit ⁵⁵ ,' apparently corresponding to the same word (Streitkräfte).
9297, 9407, 9751, 9828	The coordination structure is odd, with conjunctions (<code>coord</code>) that have a single child node, and links lower down that go across it
9360, 9602	No token numbered zero
9444	The personal pronoun 'es' is marked as plural, not singular
9573	The personal pronoun 'meine' is marked as plural, not singular
9774	Directed cycle (obj/sb/mo/pd)

in the dependency bank representation of the sentence. An example⁸ of the output of this first step, an intermediate representation with ambiguous tokens, is shown below. There are ten separate tokens, and one of them (1013136) is symbolic: the word in the sentence is ‘da’ but the representation in the dependency bank is ‘pro.’ This token is either the first or the sixth. The token pairs four-five and six-seven are both ‘multi-words.’

```
da/2 | da/1013136[pro] [pron_type=demon]
sind/3
auch/4
Filet/5001 + stücke/5
da/2 | da/1013136[pro] [pron_type=demon]
bei/6
weiß/0
ein/10
Experte/11
```

The second step of the conversion is a scoring procedure that computes a cost for each ambiguous token, based on the distance of the dependency links. In the example, both candidates (tokens 2 & 1013136) have only a single link, to the next word in the sentence in each case. Counting the distances, in tokens, of those links in all four possible cases, gives this result.

	sind/ (token 2)	bei (token 7)
da (token 1)	1 token	6 tokens
da (token 6)	-4 tokens	1 token

In this simple case, it is obvious that the minimum total (absolute) link length is achieved with the following ordering of the formerly ambiguous tokens.

```
da/2
da/1013136[pro] [pron_type=demon]
```

The disambiguated representation of the sentence is then the following.

```
da/2 sind/3 auch/4 Filet/5001 + stücke/5
da/1013136[pro] [pron_type=demon] bei/6 weiß/0 ein/10 Experte/11
```

In the (more common) situation where there are multiple dependency links to and from the ambiguous tokens, the average length of the links per word is used instead (and some weights are added). Though the total ambiguity is apparently lower in the TiGer dependency bank than in PARC 700, the frequency of highly ambiguous sentences seems higher, mainly because of the high number of (similar) articles in German. During the conversion of PARC 700 (By, 2007, p. 269) only three sentences were too difficult or time-consuming for the automatic procedure, and had to be hard-coded. For the TiGer depbank, this number was about fifty (out of three times as many sentences). In about thirty cases, sentences typically containing between five and ten articles each, the software failed to produce a

⁸Sentence 8002, the second in the TiGer dependency bank.

result in a reasonable time, so the disambiguation was hard-coded. In another hundred or so, the resulting disambiguation was found to be wrong, so these were also corrected by hand.

A couple of methods were used to verify the disambiguations, by processing the Prolog data. Simple Prolog code was written to check that all pronouns had the right person, number, and case attributes, and to compare the word order with the numerical order of the index numbers. These numbers (the value of the ‘`index`’ attribute in the Prolog version) comes originally from the TiGer treebank, where all tokens are numbered consecutively starting at ‘one.’ Because the root of the dependency tree in the depbank must have index ‘zero,’⁹ and because new numbers must be introduced for components of ‘multi-words,’ and perhaps for further reasons not known to the present author, the index numbers in the depbank are not always consecutive. Sometimes there are indexes that are higher than the number of tokens in the sentence.¹⁰ In about fifty cases there are single tokens that are not in order, and in some sentences,¹¹ the index order is apparently random.

7 The Prolog format

The BNF of the Prolog representation is shown in figure 7, and figure 5 shows an example sentence as a graphical tree, while figure 6 lists the PARC 700 and Prolog versions of it side by side. The advantages of the Prolog format is that the tokens have exactly the same sequence of alphabetical characters¹² as in the original text; that the word order is encoded; that there is no artificial distinction between full tokens and attribute tokens, and a clear division between words and empty nodes; and, finally, that the attributes are stored in one place together with the token and not spread out individually. Additionally, data in Prolog format can be loaded into a Prolog interpreter database with a single command, and then explored very easily by querying the database, or printed in other formats using short bits of code. This combination of human-friendly syntax for data and an interpreter with a database is unique to Prolog (and some closely related languages).

It should be noted that no linguistic transformations or modifications has happened to the data during the conversion, apart from corrections of the errors listed in table 3 (and that some unlinked words, e.g. infinitive markers, have been added).

The converted and slightly corrected TiGer dependency bank, in Prolog format, can be downloaded from the following web page.

<http://www.basun.net/homepages/tomas/papers/tiger/>

There are two files: ‘`tiger.pl`’ is the Prolog data, in the format shown in figure 7, and ‘`tiger.pdf`’ contains the graphical trees of all the 1867 sentences, in the classical format¹³ used also in figures 1–5 in this paper.

⁹For these tokens, the original Treebank index is the value of the ‘`tiger_id`’ attribute.

¹⁰e.g. 2100, 1012547, 1012546, 7100 in 9433 (fig. 3); 22 in 9572 (fig. 4).

¹¹Sentences 8057, 8100, 8123, 8140, 8168, 8172, 8175, 8211, 8215, 8288, 8307, 8332, 8422, 8451, 8526, 9217, 9275, 9311, 9502, 9620, 9744, 9836, 9852, and 9963.

¹²Punctuation is included only when it is part of the dependency structure in the original TiGer dependency bank files.

¹³e.g. Kunze (1975, p. 16); Matthews (1981, pp. 78–82).

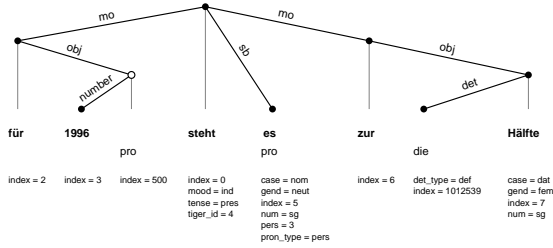


FIGURE 5: Sentence 9438 as a graphical tree

```

sentence(
  id(TiGerDB_9435)
  sentence_form("Für 1996 steht
                es zur Hälfte.")
  structure(
    mo(stehen~0, für~2)
    mo(stehen~0, zu~6)
    mood(stehen~0, ind)
    sb(stehen~0, pro~5)
    tense(stehen~0, pres)
    tiger_id(stehen~0, 4)
    obj(für~2, pro~500)
    case(pro~5, nom)
    gend(pro~5, neut)
    num(pro~5, sg)
    pers(pro~5, 3)
    pron_type(pro~5, pers)
    obj(zu~6, Hälfte~7)
    case(Hälfte~7, dat)
    det(Hälfte~7, die~1012539)
    gend(Hälfte~7, fem)
    num(Hälfte~7, sg)
    number(pro~500, 1996~3)
    det_type(die~1012539, def) ) )

sentence(0,'s9435/*9438.fdsoc',[0,1,2,3,4,5,6],[0]).
word(0,0,'für',',[index-'2']).
word(0,1,'1996',',[index-'3']).
word(0,2,steht,',', [index-'0',
                    'tiger_id'-'4',
                    'tense-pres',
                    'mood-ind']).
word(0,3,es,pro, [index-'5',
                 'gend-neut','pron_type'-pers,
                 'pers-'3','num-sg,case-nom]).
word(0,4,zu,',', [index-'6']).
word(0,5,',',die, [index-'1012539',
                  'det_type'-def]).
word(0,6,'Hälfte',',[index-'7',
                  'gend-fem,num-sg,case-dat']).
node(0,0,pro, [index-'500']).
dependency(0,w(0),mo,w(2)).
dependency(0,w(3),sb,w(2)).
dependency(0,w(4),mo,w(2)).
dependency(0,n(0),obj,w(0)).
dependency(0,w(6),obj,w(4)).
dependency(0,w(5),det,w(6)).
dependency(0,w(1),number,n(0)).

```

FIGURE 6: PARC 700 / TiGer versus Prolog representation of sentence 9438

```

Clause ::= sentence( SentNum, IdString, WordNums, NodeNums )
        | word( SentNum, WordNum, Word, Category, Attributes )
        | node( SentNum, NodeNum, Category, Attributes )
        | dependency( SentNum, Depnode, Dependency, Depnode_to )

SentNum ::= Number (Identifying the sentence)
WordNum ::= Number (Identifying the word)
NodeNum ::= Number (Identifying the empty node)
IdString ::= Atom (Arbitrary string identifying the sentence)
WordNums ::= List of numbers (In the right sentence order)
NodeNums ::= List of numbers (In arbitrary order)
Word ::= Atom
Category ::= Atom
Attributes ::= List of pairs of atoms
Depnode ::= w( WordNum ) | n( NodeNum )
Dependency ::= Atom (Name of the grammatical relation)

```

FIGURE 7: The format of the Prolog dependency representation

8 Summary

With the PARC 700 format used by the TiGer dependency bank, the combination of lemmatised tokens and the lack of encoding of the word-order means that sentences with more than one occurrence of token types with the same base form are ambiguous. The number of such ambiguous tokens is about 9% in the TiGer dependency bank.

The Prolog version presented here avoids these problems by representing the word order explicitly and using the surface forms of the words rather than the base forms. Additionally, it makes a clear distinction between words tokens and empty nodes, and stores the token attributes more compactly. The aim of this work is to make the TiGer dependency bank more accessible and easier to use.

References

- Tomas BY (2007), Some notes on the PARC 700 dependency bank, *Natural Language Engineering*, 13(3):261–282.
- John CARROLL, Ted BRISCOE, and Antonio SANFILIPPO (1998), Parser Evaluation: a Survey and a New Proposal, in *Proceedings of the first International Conference on Language Resources and Evaluation*.
- R. CROUCH, R. KAPLAN, T.H. KING, and S. RIEZLER (2002), A Comparison of Evaluation Metrics for a Broad Coverage Parser, in *Workshop on Beyond PARSEVAL at the Language Resources and Evaluation Conference*, Canary Islands, Spain.
- Martin FORST, Núria BERTOMEU, Berthold CRYSMANN, Frederik FOUVRY, Silvia HANSEN-SCHIRRA, and Valia KORDONI (2004), Towards a dependency-based gold standard for German parsers - The TiGer Dependency Bank, in *Proceedings of the COLING Workshop on Linguistically Interpreted Corpora*, Geneva.
- Haim GAIFMAN (1965), Dependency Systems and Phrase-Structure Systems, *Information and Control*, 8:304–337.
- David G. HAYS (1964), Dependency Theory: A Formalism and Some Observations, *Language*, 40(4):511–525.
- Richard HUDSON (2004), *Language Networks*, Oxford University Press, Oxford.
- Tracy Holloway KING, Richard CROUCH, Stefan RIEZLER, Mary DALRYMPLE, and Ronald M. KAPLAN (2003), The PARC 700 Dependency Bank, in *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora, held at the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*, Budapest.
- Jürgen KUNZE (1975), *Abhängigkeitsgrammatik*, Akademie-Verlag, Berlin.
- P.H. MATTHEWS (1981), *Syntax*, Cambridge University Press.
- Igor A. MEL'ČUK (1988), *Dependency Syntax: Theory and Practice*, State University of New York Press.
- Jane J. ROBINSON (1970), Dependency Structures and Transformational Rules, *Language*, 46(2):259–285.
- Lucien TESNIÈRE (1980), *Grundzüge der strukturalen Syntax*, Klett-Cotta, Stuttgart.

