ARBEITEN ZUR MEHRSPRACHIGKEIT
WORKING PAPERS IN MULTILINGUALISM

96 • 2011

Folge B • Series B

Hanna Hedeland, Thomas Schmidt, Kai Wörner (eds.)

Multilingual Resources and Multilingual Applications

Proceedings of the Conference of the
German Society for Computational Linguistics and
Language Technology (GSCL) 2011

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Sonderforschungsbereich
Mehrsprachigkeit

ARBEITEN ZUR MEHRSPRACHIGKEIT
WORKING PAPERS IN MULTILINGUALISM
Folge B, Nr. 96 • 2011

Hanna Hedeland, Thomas Schmidt, Kai Wörner (eds.)

Multilingual Resources and Multilingual Applications

Proceedings of the Conference of the German Society for Computational Linguistics and Language Technology (GSCL) 2011

Hamburger Zentrum für Sprachkorpora
Max Brauer-Allee 60
D-22765 Hamburg

Die „Arbeiten zur Mehrsprachigkeit – Folge B“ publizieren Forschungsarbeiten aus dem Sonderforschungsbereich 538 *Mehrsprachigkeit,* der von der Deutschen Forschungsgemeinschaft im Juli 1999 an der Universität Hamburg eingerichtet wurde. Wir danken der DFG für ihre Unterstützung.

Die „Arbeiten zur Mehrsprachigkeit – Folge B“ sind bei der Deutschen Bibliothek in Frankfurt/Main mit der Seriennummer ISSN 0176-559X eingetragen.

Redaktion:
Martin Elsig, Svenja Kranich, Thomas Schmidt, Manuela Schönenberger
Technische Umsetzung:
Thomas Schmidt

A German Grammar for Generation in OpenCCG

Jean Vancoppenolle* **Eric Tabbert*** **Gerlof Bouma+** **Manfred Stede***

* Dept of Linguistics, University of Potsdam, + Dept of Swedish, University of Gothenburg
E-mail: *{vancoppenolle,tabbert,stede}@uni-potsdam.de +gerlof.bouma@gu.se

Abstract

We present a freely available CCG fragment for German that is being developed for natural language generation tasks in the domain of share price statistics. It is implemented in OpenCCG, an open source Java implementation of the computationally attractive CCG formalism. Since generation requires lexical categories to have semantic representations, so that possible realizations can be produced, the underlying grammar needs to define semantics. Hybrid Logic Dependency Semantics, a logic calculus especially suited for encoding linguistic meaning, is used to declare the semantics layer. To our knowledge, related work on German CCG development has not yet focused on the semantics layer. In terms of syntax, we concentrate on aspects of German as a partially free constituent order language. Special attention is payed to scrambling, where we employ CCG's type-changing mechanism in a manner that is somewhat unusual, but that allows us to a) minimize the amount of syntactic categories that are needed to model scrambling, compared to providing categories for all possible argument orders, and b) retain enough control to impose restrictions on scrambling.

Keywords: CCG, Generation, Scrambling, German

Introduction

"Der Kurs der Post ist vom 13. September bis 29. Oktober stetig gefallen und dann bis zum 15. November wieder leicht angestiegen.

Zwischen dem 13. und dem 29. September schwankte der Kurs leicht zwischen 15 und 16 Euro. Anschließend fiel er um mehr als die Hälfte ab und erreichte am 29. Oktober seinen Tiefststand bei 7 Euro. Bis zum 15. November stieg der Kurs nach einigen Schwankungen auf seinen Schlusswert von 10 Euro."

Consider the graph depicting the development of a share price. Undoubtedly, a human could interpret the mathematical properties of that graph and quite easily describe this information in prose. He would probably produce a text more or less similar to the one presented above. In computational linguistics (or, more general, artificial intelligence), people attempt to go one step further and let the computer do that work for us. Basically, it will have to perform the same steps that a human would need to in order to accomplish this task: determine the mathematical properties of interest and generate a text that is faithful to the input and easy to read. The present paper addresses the latter sub task – i.e., the text generation.

Our goal is to develop a freely available fragment of a German grammar in OpenCCG that is suitable for natural language generation tasks in the domain of share prices. Related work on German in OpenCCG includes Hockenmaier (2006) and Hockenmaier and Young (2008), who employ grammar induction algorithms to induce CCG grammars automatically from treebanks (e.g. TiGERCorpus). To our knowledge, however, very little resources are actually freely available. In particular, the coverage of a part of the semantic layer is a novel contribution of the grammar that we present here.

1. CCG

CCG (Combinatory Categorial Grammar, Steedman 2000, Steedman & Baldridge 2011) is a lexicalized grammar formalism in which all constituents, lexical

ones included, are assigned a syntactic category that describes its combinatory possibilities. These categories may be atomic or complex. Complex categories are functions from one category into another, with specification of the relative position of the function and its argument. For instance, the notation $s\backslash np$ describes a complex category that can be combined with an np on its left (direction of the slash) to yield an s. Category combination always applies to adjacent constituents and is governed by a set of combinatory rules, of which the simplest is function application. In the example in Fig. 1, we build a sentence (category s) around a transitive verb ($(s\backslash np)/np$). There are two versions of function application used in the derivation: backward (<) and forward (>), depending on which constituent is the argument and which is the function. An overview of other derivation rules is given in Table 1.

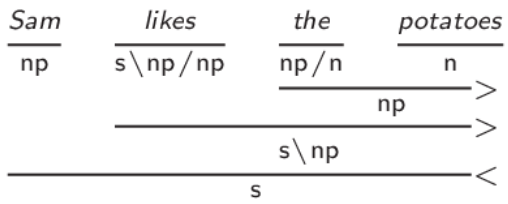

Figure 1: A basic CCG derivation.

The atomic categories in CCG come from a very restricted set. They may be enriched with features to handle case, agreement, clause types, etc. In addition, a grammar writer may choose to handle language-specific phenomena with unary type-changing rules. Finally, the grammar presented uses *multi-modal* CCG (henceforth MMCCG), which gives extended lexical control over derivation possibilities by adding modalities to the slashes in complex categories (see Baldridge 2002; Steedman & Baldridge 2011, for introduction and overview).

In its basic form, CCG has mildly context-sensitive generative power and is thus able to account for non-context-free natural language phenomena (Steedman, 2000). Its attractiveness is due to the linguistic expressiveness on the one hand and the fact that it is efficiently parsable in theory (Shanker & Weir, 1990), as well as in practice (Clark & Curran, 2007).

Function application				
$(>)$	α/β	β	$\Rightarrow$	α
$(<)$	β	$\alpha\backslash\beta$	$\Rightarrow$	α
Function composition				
$(>\mathbf{B})$	α/β	β/γ	$\Rightarrow$	α/γ
$(<\mathbf{B})$	$\beta\backslash\gamma$	$\alpha\backslash\beta$	$\Rightarrow$	$\alpha\backslash\gamma$
Crossed function composition				
$(>\mathbf{B}_\times)$	α/β	$\beta\backslash\gamma$	$\Rightarrow$	$\alpha\backslash\gamma$
$(<\mathbf{B}_\times)$	β/γ	$\alpha\backslash\beta$	$\Rightarrow$	α/γ
Type raising				
$(>\mathbf{T})$	α		$\Rightarrow$	$\beta/(\beta\backslash\alpha)$
$(<\mathbf{T})$	α		$\Rightarrow$	$\beta\backslash(\beta/\alpha)$

Table 1: Basic combinatory rules.

1.1. OpenCCG

OpenCCG[1] is an open source Java implementation of an MMCCG parser and generator. On the semantic side, it implements HLDS (Hybrid Logic Dependency Semantics, Blackburn, 2000; Baldridge & Kruijff, 2002), an extended modal logic calculus especially suited for encoding linguistic meaning. Below is an example sentence and its corresponding HLDS formula:

1) Der Kurs erreicht seinen Höchststand.
the share-price reaches its peak

$$
\begin{aligned}
&@_{e:\text{achievement}}(\mathbf{reach}\wedge\\
&\quad\langle\text{ARG1}\rangle(r:\text{sem-obj}\wedge\mathbf{rate}\wedge\\
&\qquad\langle\text{DEF}\rangle+\wedge\\
&\qquad\langle\text{NUM}\rangle\text{sg}\wedge\\
&\quad\langle\text{ARG2}\rangle(p:\text{sem-obj}\wedge\mathbf{peak}\wedge\\
&\qquad\langle\text{NUM}\rangle\text{sg}\wedge\\
&\qquad\langle\text{OWNER}\rangle(i:\text{sem-obj}\wedge\text{pro3m})))
\end{aligned}
$$

Figure 2: HLDS formula of 1).

In HLDS, the meaning of (1) is represented by means of dependency relations between discourse referents, such as the OWNER relation between the referents **peaks** and **its** (pro3m) in Fig. 2. Nominals uniquely identify discourse referents and provide a means to reference them at any point in a formula. In Fig. 2, the nominals *e*, *r*, *p*, and *i* identify the referents for the event of **reach**ing, **rate**, **peak** and **its**, respectively. Nominals can be augmented to restrict the semantic type of possible discourse referents with respect to an underlying ontology, for instance to animate or inanimate entities, or, as in Fig. 2, to a more general notion like *sem*antic

[1]http://openccg.sourceforge.net/

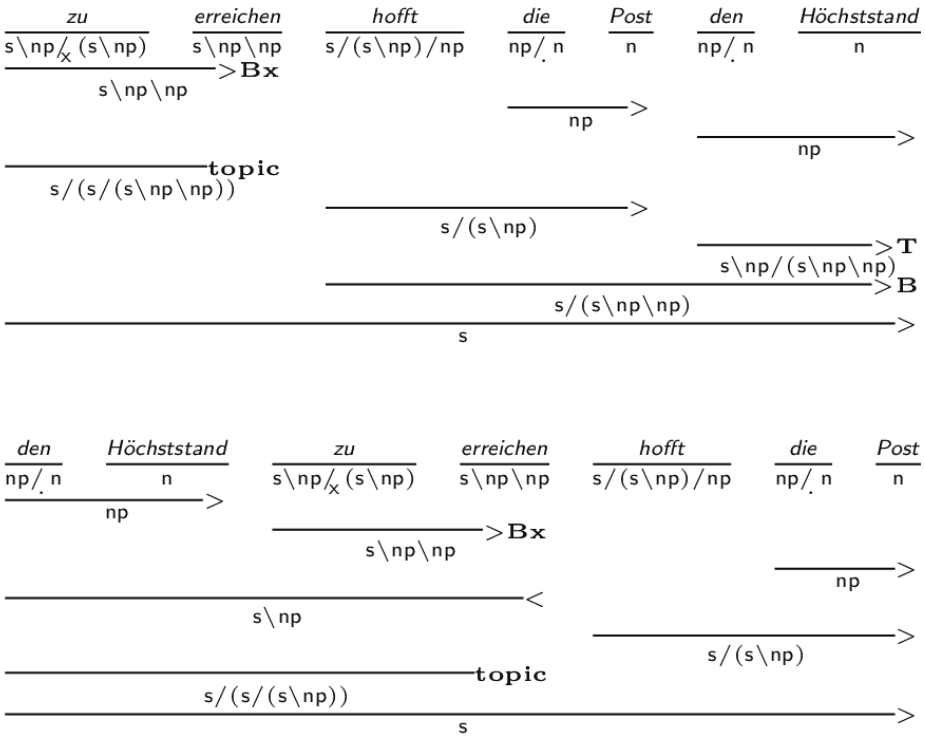

Figure 3: NP fronting

*obj*ects. Finally, the satisfaction operator $@$ states that the formula p in $@_i p$ holds at world *i*.

OpenCCG implements a flexible surface realizer that when given a logical form (LF) like in Fig. 1 returns one or more realizations of it, based on the underlying grammar. Both the number of realizations and their surface forms depend on how much information a LF specifies, thereby allowing to either enhance or restrain non-determinism of the realization process. For example, given that the LF in Fig. 2 does not specify which of the two arguments is fronted, the following two surface forms are possible:

2) Der Kurs erreicht seinen Höchststand.
the share-price reaches its peak

3) Seinen Höchststand erreicht der Kurs.
Its peak reaches the share-price.
'The share-price reaches its peak'

2. Coverage

Our current work focuses on different aspects of German as a partially free constituent order language, including basic constituent order and scrambling in particular, but also on complex nominal phrases, clausal subordination, and coordination. In the next two sections, we first give a brief overview of how topicalization is modeled in our grammar, followed by an approach to scrambling that we are currently investigating and that, as far as we know, is new in CCG.

2.1. Topicalization

The finite verb can occupy three different positions that depend on the clause type and determine the sentence mood: matrix clauses are either verb-initial (declarative or yes/no-interrogative), or verb-second (declarative or wh-interrogative), and subordinate clauses are always verb-final (declarative or interrogative).

Following Steedman (2000), Hockenmaier (2006) and Hockenmaier and Young (2008), we implemented a topicalization rule that systematically derives verb-second order from verb-initial order by fronting an argument of the verb, e.g. an NP, a PP, or a clause. This also covers partial fronting (see Fig. 3 for examples):

4) $T \Rightarrow s_{v2}/(s_{v1}/T)$, $T = \{np, pp, s_{to\text{-}inf}\backslash np, ...\}$

Sentence modifiers (e.g. *heute* in *heute fällt der Kurs* 'today, the share price is falling') are analyzed as s_{v2}/s_{v1} and can thus form verb-second clauses on their own.

2.2. Scrambling

Much of the constituent order freedom in German is due to the fact that it allows for permutation of verbal arguments within a clause (local scrambling, 5) and 'extraction' of arguments of an arbitrarily deeply embedded infinite clause (long-distance scrambling, 6):

5) dass [dem Unternehmen]$_2$ [das Richtige]$_3$
that the enterprise the right-thing
[der Berater]$_1$ empfiehlt.
the counselor advises
'that the counselor advises the enterprise the right thing'

6) dass [dem Unternehmen]$_2$ [das Richtige]$_3$
that the enterprise the right-thing
[der Berater]$_1$ [_ _ zu empfehlen hofft].
the counselor to advise hopes
'that the counselor hopes to advise the enterprise the right thing'

Different proposals have been made in MMCCG to account for constituent order freedom in general. To our knowledge, the two most common approaches are to provide separate categories for each possible order (Hockenmaier, 2006; Hockenmaier & Young, 2008) or to allow lexical underspecification of argument order through multi-sets (Hoffman, 1992; Steedman & Baldridge, 2003).

We are investigating an approach to local scrambling that aims at combining the advantages of both methods, namely having fine-grained control over argument permutation on the one hand, and requiring as few categories as possible on the other. It is based on a set of type-changing rules that change categories 'on the fly'. (7) shows a simplified rule that allows to derive plural NPs from plural nouns, reflecting the optionality of determiners in German plural NPs (e.g. *sie isst Kartoffeln* 'she eats potatoes'):

7) $n_{pl} \Rightarrow np_{pl}$

Type-changing rules can also be used to swap two consecutive argument NPs, (*i* and *j* denote indexes):

8) $s/np_{\langle i\rangle\text{+base}}/np_{\langle j\rangle\text{-pron}} \Rightarrow s/np_{\langle j\rangle}/np_{\langle \sim i\rangle\text{-base}}$

9) $s\$\backslash np_{\langle i\rangle\text{-pron}}\backslash np_{\langle j\rangle\text{+base, -pron}} \Rightarrow s\$\backslash np_{\langle \sim j\rangle\text{-base, -pron}}\backslash np_{\langle i\rangle}$

This essentially emulates the behavior of multi-sets and at the same time reduces the number of categories to a minimum, thereby enhancing the maintainability of the grammar. The advantage over multi-sets is that restrictions on scrambling can be formulated straightforwardly, such as that full NPs should not scramble over pronouns (i.e. NPs having the *-pron*(oun) feature) (see Uszkoreit (1987) for an overview of scrambling regularities in German).

Rules like (8) and (9) require special caution, though. Type-changing rules are supposed to actually *change* the type of the argument category as they could otherwise apply over and over again, causing an infinite recursion. This is where the ±*base* feature comes into play. It indicates whether an NP occupies its base position or has already been scrambled, restricting the application of (8) and (9) to the former case and thereby preventing infinite recursion. The so-called *dollar variable* \$ in (9) ranges over complex categories that have the same functor (here: s), such as s\np . It is not crucial to our scrambling rules but generalizes (9) to apply to both transitive and ditransitive verbs.

Four more rules are sufficient to capture all possible local permutations and also some of the long-distance permutations, as the one in (6).

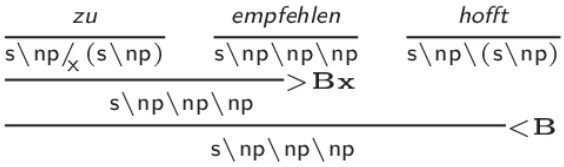

Figure 4: Parse of an infinite clause.

The derivation in Fig. 4 contains the derivation of the complex verb cluster of example (6). The composed category s\np\np\np corresponds to the one of an ordinary ditransitive verb, so although (6) is an instance of long-distance scrambling, it can be derived by means of our local scrambling rules (8) and (9).

3. Lexicon

The grammar is intended for use in the domain of the stock market, thus providing the means to describe the development of share prices. Since the expansion and proper implementation of a lexical database is a full-fledged task of its own and the focus of our current work is to extend the grammar, our current lexicon is still quite limited in its scope.

At a later point one might consider to make use of the

CCGbank lexicon (Hockenmaier, 2006).

3.1. Nouns

Our lexicon currently contains approximately 125 nouns. For the different inflectional paradigms we made use of inflection tables presented on the free online service canoo.net.[2] For each of these paradigms we wrote an 'expansion'. OpenCCG's expansions provide a means to define inflectional paradigms as an applicable rule and link lexical information to them, so that OpenCCG generates the different tokens of a word and its syntactic and semantic properties as interpretable lexical entries. Thus a typical noun entry is a one-liner like this:

10) # Höchststand
noun_infl_1(Höchststand, Höchstständ, masc peak, graph_point_definition)

The first two arguments contain the singular and plural stem, to which the inflection endings will be attached by the expansion. The following arguments are gender (for agreement), a predicate (as semantic reference) and a semantic type from the ontology. While seemingly plain English, these semantic predicates should be thought of as grossly simplified meta language, which guarantees a unique and unambiguous semantic representation.

3.2. Verbs

For the verbs we followed a similar approach, with three expansions. The first two actually cover the same inflection paradigm, with the difference that for verbs ending in -ern like *klettern* (to climb) we duplicated the paradigm and made slight adjustments to circumvent the concatenation of the word stem *kletter* and certain inflectional morphs like *-en* to ungrammatical forms like **(wir) kletteren* (instead of *klettern*). The third expansion covers several modal verbs like *können* (to can) or *müssen* (to have to).

Each of those rules sets the features of the respective inflection (e.g. fin, 1st, sg, pres) and those for past tense. Sample entries:

11) regular-vv(schwanken, schwank, schwankte, fluctuate)
12) regular-vv-ern(klettern, kletter, kletterte, climb)

[2]http://www.canoo.net/

4. Generation

We would like to conclude with a brief outline of how our grammar fits into the generation scenario presented in the introduction.

The idea is to generate text automatically from share price graphs, i.e., from collections of data points. Graphs are analyzed in terms of different mathematical properties (e.g. extremes and inflection points). These properties, together with user-provided realization parameters that allow fine-grained control over the 'specificity' of LFs (and thus over the number of surface realizations), are input to static LF templates. The filled LF templates are then fed to the OpenCCG realizer where our grammar is used to compute the appropriate surface forms. In the last step, orthographic post-processing, the surface forms are normalized with respect to language-specific orthographic standards (e.g. number or date formats, etc.). The figure below illustrates this procedure:

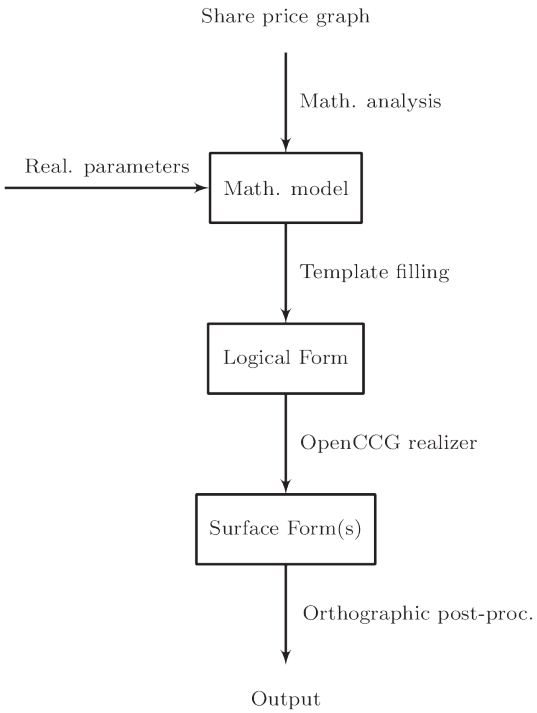

Figure 5: Procedure of the generation process.

5. Summary

We have presented a freely available CCG fragment of a generation grammar for German that is equipped with a semantic layer implemented in Hybrid Logic

Dependency Semantics. In terms of syntax, we have focused on aspects of German as a partially free constituent order language and investigated an approach to scrambling by employing OpenCCG's type-changing rules in a somewhat unconventional manner. In doing so, we aimed at minimizing the amount of categories needed to allow different argument orders while retaining a certain degree of flexibility regarding argument order restrictions. Future work will concentrate more on the lexicon, for instance by refining and extending our expansions for inflectional paradigms of various word classes. We also hope to use OpenCCG's interesting regular expression facilities for derivational morphology.

Our grammar can be downloaded from www.ling.uni-potsdam.de/~stede/AGacl/ressourcen/GerGenGram.

6. Acknowledgements

We would like to thank the other participants of the course *Automatische Textgenerierung* (Winter 2010/11) at the University of Potsdam, and also the GSCL reviewers for their comments.

7. References

Baldridge, J. (2002): Lexically Specified Derivational Control in Combinatory Categorial Grammar. PhD thesis, School of Informatics, University of Edinburgh.

Baldridge, J., Kruijff, G.-J.M. (2002): Coupling CCG and Hybrid Logic Dependency Semantics. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002).

Baldridge, J., Kruijff, G.-J.M. (2003): Multi-Modal Combinatory Categorial Grammar. In Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2003).

Blackburn, P. (1993): Modal Logic and Attribute Value Structures. In M. de Rijke, editors, Diamonds and Defaults, Synthese Language Library, pp. 19–65, Kluwer Academic Publishers, Dordrecht, 1993.

Blackburn, P. (2000): Representation, Reasoning, and Relational Structures: a Hybrid Logic Manifesto. Logic Journal of the IGPL, 8(3), pp. 339-625.

Bozsahin, C., Kruijff, G.-J.M., White, M. (2008): Specifying Grammars for OpenCCG: A Rough Guide. http://openccg.sourceforge.net/

Clark, S., Curran, S. (2007): Wide-coverage Efficient Statistical Parsing with CCG and Log-linear Models. Computational Linguistics, 33(4), pp. 493-552.

Drach, E. (1937): Grundgedanken der deutschen Satzlehre. Diesterweg.

Hockenmaier, J. (2006): Creating a CCGbank and a wide-coverage CCG lexicon for German. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL.

Hockenmaier, J., Young, P. (2008): Non-local scrambling: the equivalence of TAG and CCG revisited. Proceedings of The Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms, pp. 41–48, Tübingen, Germany.

Hoffman, B. (1992): A CCG Approach to Free Word Order Languages. Proceedings of the 30th Annual Meeting of ACL, pp. 300-302.

Müller, S. (2010): Grammatiktheorie. Stauffenburg Verlag.

Steedman, M. (2000): The Syntactic Process. MIT Press.

Steedman, M., Baldridge, J. (2011): Combinatory Categorial Grammar. In Borsley and Börjars (eds), Non-tranformational Syntax: Formal and explicit models of grammar, Wiley-Blackwell.

Uszkoreit, H. (1987): Word Order and Constituent Structure in German. CSLI.

Vijay-Shanker, K., Weir, D.J. (1990): Polynomial Time Parsing of Combinatory Categorial Grammars. Proceedings of the 28th Annual Meeting of Computational Linguistics, pp. 1-8, Pittsburgh, PA, June 1990.

White, M.: OpenCCG Realizer Manual. Documentation of the OpenCCG Realizer.

White, M. (2004): Efficient Realization of Coordinate Structures in Combinatory Categorial Grammar. Research on Language & Computation, 4(1), pp. 39-75.

White, M., Rajkumar R., Martin, S. (2007): Towards Broad Coverage Surface Realization with CCG. In Proceedings of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+MT).