

# Glue Rules for Robust Chart Realization

Michael White

Department of Linguistics

The Ohio State University

Columbus, Ohio, USA

mwhite@ling.ohio-state.edu

## Abstract

This paper shows how glue rules can be used to increase the robustness of statistical chart realization in a manner inspired by dependency realization. Unlike the use of glue rules in MT—but like previous work with XLE on improving robustness with hand-crafted grammars—they are invoked here as a fall-back option when no grammatically complete realization can be found. The method works with Combinatory Categorical Grammar (CCG) and has been implemented in OpenCCG. As the techniques are not overly tied to CCG, they are expected to be applicable to other grammar-based chart realizers where robustness is a common problem. Unlike an earlier robustness technique of greedily assembling fragments, glue rules enable  $n$ -best outputs and are compatible with disjunctive inputs. Experimental results indicate that glue rules yield improved realizations in comparison to greedy fragment assembly, though a sizeable gap remains between the quality of grammatically complete realizations and fragmentary ones.

## 1 Introduction

Robustness continues to be a problem for broad coverage chart realizers. Since Kay’s (1996) pioneering work on chart realization with unification grammars, broad coverage chart realizers have been developed for LFG (Shemtov, 1997; Cahill and van Genabith, 2006; Hogan et al., 2007), HPSG (Velldal et al., 2004; Nakanishi et al., 2005; Velldal and Oepen, 2005; Carroll and Oepen, 2005) and CCG (White,

2006b; White, 2006a; Espinosa et al., 2008; White and Rajkumar, 2009), but none of these realizers come near 100% coverage. For example, both Cahill and van Genabith (2006) and White and Rajkumar (2009) report coverage below 90% for all Penn Treebank test section sentences (despite coverage near 100% for parsers with comparable grammars), and consequently both also report results with fragment concatenation for increased robustness. Earlier work with hand-crafted grammars for the XLE realizer has also made it possible to specify fragment concatenation rules.<sup>1</sup> Failure to generate a grammatically complete realization can be expected to become an even greater issue in surface realization shared tasks, where realizers must cope with non-native “common ground” inputs.

In contrast to grammar-based chart realization approaches, recent dependency-based approaches (Guo et al., 2008; Gali and Venkatapathy, 2009; Guo et al., 2010), which have eschewed explicit grammatical constraints, easily achieve 100% coverage by simply ensuring that each input word in a dependency structure ends up in the output. As the adage goes, if you can’t beat ’em, join ’em, and thus in this paper we take a step in this direction by investigating the use of MT-inspired glue rules (Chiang, 2007) for enhanced robustness. The idea is that by using glue rules as a fall-back option, in the limit chart realization simply degenerates into dependency realization. The catch, of course—beyond computational concerns—is that in unmodified form, realization ranking models for grammar-based realiza-

<sup>1</sup><http://www2.parc.com/isl/groups/nlitt/xle/doc/xle.html#SECG5>

tion are unlikely to work as well as ones designed explicitly for dependency-based realization.

Our approach to employing glue rules in chart realization is cached out in Combinatory Categorical Grammar (Steedman, 2000, CCG) and implemented in OpenCCG,<sup>2</sup> though as the techniques are not overly tied to CCG, we expect them to be applicable to other grammatical frameworks as well. To date, OpenCCG has made use of a greedy approach to assembling fragments when no grammatically realization is found within a time limit, which starts with the largest fragment and greedily adds non-overlapping fragments to one end or the other in a way that locally maximizes the realization ranking model score. In comparison to this earlier method, glue rules enable a much larger space of fragment concatenations to be explored, and since these rules are integrated into the general chart realization framework, they remain compatible with returning  $n$ -best outputs and allowing disjunctively specified inputs, in contrast to the earlier greedy concatenation method.<sup>3</sup>

## 2 Background

OpenCCG is a parsing/generation library for CCG which includes a hybrid symbolic-statistical chart realizer (White, 2006b). The chart realizer takes as input (quasi-) logical forms (LFs) represented using Hybrid Logic Dependency Semantics (HLDS), a dependency-based approach to representing linguistic meaning (Baldrige and Kruijff, 2002); see White (2006b) for discussion. Semantic dependency graphs are derived from the CCGbank (Hockenmaier and Steedman, 2007), modified to incorporate Propbank roles (Boxwell and White, 2008), where semantically empty function words such as complementizers, relativizers, infinitival-*to*, and expletive subjects are adjusted to reflect their purely syntactic status. Lexical category assignments are statistically filtered in a hypertagging step (Espinosa et

<sup>2</sup><http://openccg.sf.net>

<sup>3</sup>While the greedy approach to fragment assembly could conceivably be generalized to a beam search that respected disjunctive constraints, doing so would introduce considerable redundancy with the core chart realization algorithm; indeed, generalizing the greedy approach by reusing the existing chart realization algorithm is essentially what the glue rules are designed to do.

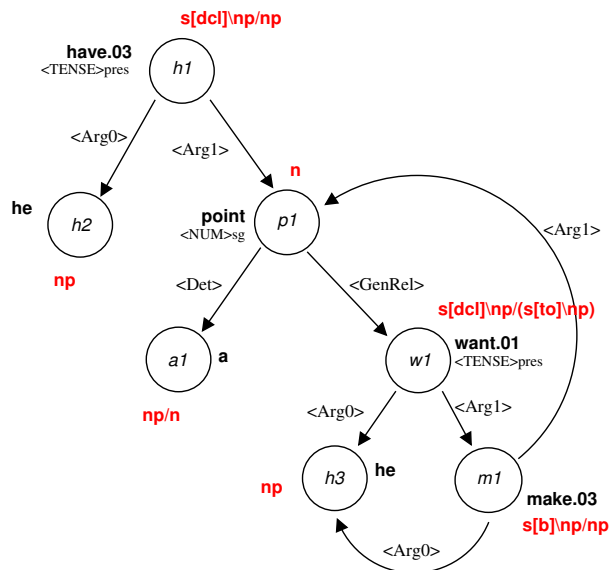


Figure 1: Semantic dependency graph from the CCGbank for *He has a point he wants to make [...]*, along with gold-standard supertags (category labels)

al., 2008); Figure 1 illustrates the desired output of the hypertagger. As in Clark & Curran’s (2007) approach to integrating supertagging and parsing, an adaptive strategy is employed, whereby a  $\beta$ -best list of supertags is returned for each lexical predication, and the hypertagger’s  $\beta$  setting is progressively relaxed until a complete realization is found or the space/time limits are exceeded. Alternative realizations are ranked using an averaged perceptron model (White and Rajkumar, 2009) that makes use of three kinds of features: (1) the log probability of the candidate realization’s word sequence according to a trigram word model and a factored language model over part-of-speech tags and supertags; (2) integer-valued syntactic features, representing counts of occurrences in a derivation, from Clark & Curran’s normal form model; and (3) discriminative  $n$ -gram features (Roark et al., 2004), which count the occurrences of each  $n$ -gram in the word sequence. Section 4 of this paper also explores the use of a basic dependency model, with head-dependent and sibling dependent ordering features.

## 3 Glue Rules for Chart Realization

As in Chiang’s (2007) approach to using glue rules in synchronous context-free grammars and the XLE approach to fragment rules in hand-crafted gram-

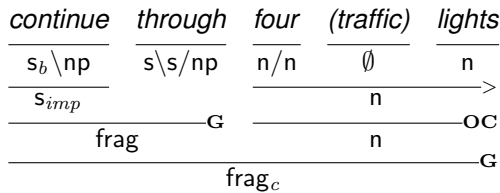


Figure 2: Syntactic derivation for *continue through four lights* using the glue rule (**G**) and opt-completion rule (**OC**), where *traffic* is left out for lack of a matching category, and *four lights* cannot be promoted to an NP because of a missing determiner semantic feature in the input.

As the example derivation (discussed further below) in Figure 2 shows, the glue rule (**G**) here is  $X Y[-frag] \Rightarrow frag$ , where any two categories can be combined into a fragment category—except that only the left category may itself be a fragment, to avoid spurious ambiguities in how fragments are concatenated.

There are three twists to this basic story. First, on the assumption that derivations that follow the grammar are to be preferred to ones employing the glue rule, glue rules are only invoked after the chart has been completed with no grammatically complete derivation found to cover the input, and then only when the glue rule fills in an empty cell (i.e. set of covered elementary predications, or EPs). Additionally, to aid in the search for a fragment that covers the input completely, edges on the realizer’s agenda are sorted first by the number of covered EPs, and secondarily by their model score.

The second twist concerns the LF chunking constraints in the realizer. In order to address the problem of proliferating semantically incomplete constituents (Kay, 1996), OpenCCG requires all the EPs in an LF chunk—by default, a non-trivial subtree in the input—to be covered by an edge before combination is allowed with another edge with EPs outside the chunk (White, 2006b). To effectively relax these constraints, if there are elementary predications within an LF chunk which are not covered by any lexical items or instantiated unary rules, those EPs are made optional; similarly, the EPs for instantiated unary rules are made optional, so that they can

```
Input LF:
@c(continue ^
  <Actor>(p ^ pro2) ^
  <Path>(t1 ^ through ^
    <Ref>(l ^ light ^ <num>pl ^
      <Card>(f ^ four) ^
      <Mod>(t2 ^ traffic))))
```

```
Preds:
ep[0]: @p(pro2)
ep[1]: @c(continue)
ep[2]: @c(<Actor>p)
ep[3]: @c(<Path>t1)
ep[4]: @t1(through)
ep[5]: @t1(<Ref>l)
ep[6]: @f(four)
ep[7]: @t2(traffic)
ep[8]: @l(light)
ep[9]: @l(<num>pl)
ep[10]: @l(<Card>f)
ep[11]: @l(<Mod>t2)
```

```
LF chunks:
chunk[0]: {6-11}
chunk[1]: {4-11}
chunk[2]: {0-11}
```

```
LF optional parts:
opt[0]: {0}
opt[1]: {7,11}
```

Figure 3: Broken HLDS LF input for *continue through four traffic lights*, where *traffic* is given with the wrong relation and the determiner feature is missing. The elementary predications (EPs) for *traffic* are made optional, for lack of a matching category, and the EP for the implicit *you*, introduced by a unary rule, is also made optional. The sub-tree chunks for *four traffic lights*, *through four traffic lights* and all the entire input are also shown.

be checked off as covered by relevant fragments.<sup>4</sup>

As a third and final twist, to allow glue rules to be applied recursively, fragments that complete an LF chunk or disjunction are marked as completed fragments ( $frag_c$ ), so that they may be used with the glue rule as the right category (where fragments are normally disallowed). Note that it is the recursive use of glue rules, along with the connection to dependency realization discussed next, that perhaps most distinguishes the present approach from the use of fragment rules in hand-crafted grammars with XLE.

<sup>4</sup>Experiments with relaxed relation matching, which is similar to the use of relaxed unification constraints in grammar-based error detection (Schwind, 1988), have been inconclusive to date. In future work, it would be interesting to further explore the use of constraint relaxation and possibly other techniques from error detection, such as the use of mal-rules (Schneider and McCoy, 1998).

As glue rules are applied, LF chunking constraints are applied as usual, and thus the fragment gluing phase becomes tantamount to exploring different permutations of heads and phrases headed by their dependents, much as in dependency-based realization approaches. That is, since fragment edges are constructed by assembling existing edges in either order, all permutations of edges whose EPs fall within an LF chunk will eventually be tried (subject so search constraints), with preference given to the orderings with the best model scores. Note that with glue rules, tracking of disjunctive alternatives and optional EPs continues as usual too, so that  $n$ -best generation and realization from disjunctive logical forms can remain enabled.

To illustrate how glue rules enhance robustness, consider the input for the derivation in Figure 2 given in Figure 3, which shows a broken LF input for *continue through four traffic lights* using OpenCCG’s `routes` sample grammar. Here, *traffic* is specified using the `Mod` relation instead of the `HasProp` relation required by the grammar, and the semantic feature for a zero determiner has been left out. Nevertheless, the realizer is able to generate *continue through four lights*, as follows. Initially, a nominal constituent (n) *four lights* is derived using the forward application rule, and the unary rule for promoting a bare verb phrase to an imperative sentence (`simp`) is applied to *continue*. As no further constituents can be formed, glue rules are enabled. At this point, *continue* and *through* combine via the glue rule (with `X` instantiated to `simp` and `Y` instantiated to `s\s/np`), and the opt-completion rule (`OC`) is invoked so that *four lights* can be considered to cover the now optional EPs for *traffic* as well.<sup>5</sup> Finally, *continue through* and *four lights* combine via the glue rule to cover all the input EPs, making a completed fragment (`fragc`).<sup>6</sup> If this clause were embedded in a larger sentence—e.g., *he said continue through four lights*—the completed fragment could again combine via the glue rule with *he said* to form a complete sentence.

<sup>5</sup>That is, since EPs 7 and 11 are optional, the edge for *four lights* can be promoted to one that covers all of the EPs 6–11 (White, 2006a).

<sup>6</sup>In  $n$ -best generation, other variants are generated as well, such as *you continue through four lights*, *continue through four lights you*, etc.

	perceptron –deps	perceptron +deps	oracle +deps
all: greedy	0.8133	0.8237	0.9409
all: glue rules	0.8198	<b>0.8308</b>	0.9570
gramm. complete	0.8686	0.8795	0.9747
greedy fragments	0.6039	0.6170	0.8158
glued fragments	0.6408	<b>0.6523</b>	0.8924

Table 2: Development set BLEU scores, CCGbank Section 00 (1575 grammatically complete sentences; 322 fragmentary ones)

	perceptron +deps
all incl. greedy fragments	0.8402
all incl. glue rule fragments	<b>0.8462</b>
grammatically complete	0.8879
greedy fragments	0.6116
glue rule fragments	<b>0.6477</b>

Table 3: Test set BLEU scores, CCGbank Section 23 (1932 grammatically complete sentences; 328 fragmentary ones)

## 4 Experimental Results

To further explore the connection to dependency realization, the dependency features illustrated in Table 1 were added to the baseline averaged perceptron realization ranking model.<sup>7</sup> These features, which depend on the input LF and candidate realization but not the CCG categories, count the occurrences of head-dependent and sibling dependent ordering configurations in a derivation. The features listed at the top record whether the head precedes the dependent or vice-versa, grouped by the broad part of speech (POS) of the head and the relation between the head and the dependent, with different combinations of words and POS tags. The features at the bottom record the order of sibling dependent words appearing on the same side of the head word, similarly grouped by the broad POS of the head and at different granularities of word or POS tag, and additionally with relation-relation orderings.

Table 2 shows the results of reverse realization with OpenCCG on the development section of the

<sup>7</sup>Features incorporating named entity classes (Rajkumar et al., 2009) and targeting agreement errors (Rajkumar and White, 2010) were not used in the experiments reported here.

Feature Type	Example
HeadBroadPos + Rel + Precedes + HeadWord + DepWord	⟨VB, Arg0, dep, wants, he⟩
... + HeadWord + DepPOS	⟨VB, Arg0, dep, wants, PRP⟩
... + HeadPOS + DepWord	⟨VB, Arg0, dep, VBZ, he⟩
... + HeadWord + DepPOS	⟨VB, Arg0, dep, VBZ, PRP⟩
HeadBroadPos + Side + DepWord1 + DepWord2	⟨NN, left, an, important⟩
... + DepWord1 + DepPOS2	⟨NN, left, an, JJ⟩
... + DepPOS1 + DepWord2	⟨NN, left, DT, important⟩
... + DepPOS1 + DepPOS2	⟨NN, left, DT, JJ⟩
... + Rel1 + Rel2	⟨NN, left, Det, Mod⟩

Table 1: Basic head-dependent and sibling dependent ordering features

CCGbank, Section 00, using a perceptron model with and without the dependency features as well as an oracle model using an n-gram precision score (approximating BLEU) against the reference sentence, which provides a topline result. Grammatically complete realizations were found for 83% of the development sentences within a 15-second time limit; in the remaining cases, outputs were constructed from the current chart either using the glue rules or the earlier greedy fragment assembly. With the glue rules, the realizer was run with packing enabled with a new 15-second limit, and complete edges were unpacked; with greedy fragment assembly, the realizer was run in best-first mode up to the new time limit, and then the available edges were greedily assembled. As the table shows, on the fragmentary cases the glue rules yield more than a three and a half point improvement in BLEU scores over greedy fragment assembly when using the perceptron scorer, both with the dependency features (from 0.6170 for 0.6523) and without them (from 0.6039 to 0.6408), showing that the modeling benefit of the dependency features carries over to the fragmentary cases. With the oracle scorer, the improvement is over 7.5 BLEU points, indicating that the glue rules may be capable of yielding even larger improvements with better ranking models.

Table 3 confirms the results of the averaged perceptron model with dependency features on the test section of the CCGbank, Section 23. As is evident in the table, the gap between the BLEU scores for the grammatically complete sentences and the fragmentary ones is quite large (more than 20 BLEU points). Thus, although the overall improvement in BLEU scores is modest (0.6-0.7 of a BLEU point) since the glue rules apply in only 15-17% of the cases,

their effect is clearly noticeable with these sentences where the outputs remain generally mediocre.

## 5 Conclusions and Future Work

This paper has shown how glue rules can be used in OpenCCG as a fall-back option when no grammatically complete realization can be found, thereby increasing the robustness of chart realization. Unlike an earlier robustness technique of greedily assembling fragments, glue rules enable  $n$ -best outputs, are compatible with disjunctive inputs, and explore a larger space of possible fragment concatenations. They also differ from the fragment concatenation rules used in hand-crafted grammars for the XLE realizer in applying recursively, enabling the glue rules to emulate dependency realization. The experimental results indicate that by enabling this larger space of assembled fragments to be explored, glue rules can yield improved realizations in comparison to greedy fragment assembly, though a sizeable gap remains between the quality of grammatically complete realizations and fragmentary ones.

In future work, we plan to experiment with realization ranking models incorporating richer dependency-based features, with the aim of further reducing the quality gap between grammatically complete and fragmentary realizations. We also plan to examine the impact of such models and the glue rules on Generation Challenges shared task results.

## Acknowledgments

This work was supported in part by NSF grant number IIS-0812297. Thanks go to the anonymous reviewers for helpful comments and discussion.

## References

- Jason Baldridge and Geert-Jan Kruijff. 2002. Coupling CCG and Hybrid Logic Dependency Semantics. In *Proc. ACL-02*.
- Stephen Boxwell and Michael White. 2008. Projecting Propbank roles onto the CCGbank. In *Proc. LREC-08*.
- Aoife Cahill and Josef van Genabith. 2006. Robust PCFG-based generation using automatically acquired LFG approximations. In *Proc. COLING-ACL '06*.
- John Carroll and Stefan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *Proc. IJCNLP-05*.
- David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228, June.
- Stephen Clark and James R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552.
- Dominic Espinosa, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with CCG. In *Proceedings of ACL-08: HLT*, pages 183–191, Columbus, Ohio, June. Association for Computational Linguistics.
- Karthik Gali and Sriram Venkatapathy. 2009. Sentence realisation from bag of words with dependency constraints. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium, NAACL '09*, pages 19–24, Morristown, NJ, USA. Association for Computational Linguistics.
- Yuqing Guo, Josef van Genabith, and Haifeng Wang. 2008. Dependency-based n-gram models for general purpose sentence realisation. In *Proc. COLING-08*.
- Yuqing Guo, Haifeng Wang, and Josef van Genabith. 2010. A linguistically inspired statistical model for chinese punctuation generation. *ACM Transactions on Asian Language Information Processing*, 9:6:1–6:27, June.
- Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.
- Deirdre Hogan, Conor Cafferkey, Aoife Cahill, and Josef van Genabith. 2007. Exploiting multi-word units in history-based probabilistic generation. In *Proc. EMNLP-CoNLL*.
- Martin Kay. 1996. Chart generation. In *Proc. ACL-96*.
- Hiroko Nakanishi, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic methods for disambiguation of an HPSG-based chart generator. In *Proc. IWPT-05*.
- Rajakrishnan Rajkumar and Michael White. 2010. Designing agreement features for realization ranking. In *Coling 2010: Posters*, pages 1032–1040, Beijing, China, August. Coling 2010 Organizing Committee.
- Rajakrishnan Rajkumar, Michael White, and Dominic Espinosa. 2009. Exploiting named entity classes in ccg surface realization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 161–164, Boulder, Colorado, June. Association for Computational Linguistics.
- Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. 2004. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proc. ACL-04*.
- David Schneider and Kathleen F. McCoy. 1998. Recognizing syntactic errors in the writing of second language learners. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 1198–1204, Montreal, Quebec, Canada, August. Association for Computational Linguistics.
- Camilla Schwind. 1988. Sensitive parsing: error analysis and explanation in an intelligent language tutoring system. In *Proceedings of the 12th Conference on Computational Linguistics*, pages 608–613.
- Hadar Shemtov. 1997. *Ambiguity Management in Natural Language Generation*. Ph.D. thesis, Stanford University.
- Mark Steedman. 2000. *The Syntactic Process*. MIT Press.
- Erik Velldal and Stefan Oepen. 2005. Maximum entropy models for realization ranking. In *Proc. MT-Summit X*.
- Erik Velldal, Stephan Oepen, and Dan Flickinger. 2004. Paraphrasing treebanks for stochastic realization ranking. In *Proceedings of the 3rd Workshop on Treebanks and Linguistic Theories*.
- Michael White and Rajakrishnan Rajkumar. 2009. Perceptron reranking for CCG realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 410–419, Singapore, August. Association for Computational Linguistics.
- Michael White. 2006a. CCG chart realization from disjunctive logical forms. In *Proc. INLG-06*.
- Michael White. 2006b. Efficient Realization of Coordinate Structures in Combinatory Categorical Grammar. *Research on Language & Computation*, 4(1):39–75.