

## SUMMARY

- **Goal:** Show how glue rules can be used to increase the robustness of statistical chart realization, in a manner inspired by dependency realization
- **Comparison:** In contrast to 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 complete realization can be found
- **Benefits:** Unlike an earlier technique of greedily assembling fragments, glue rules are well-integrated into the chart-based search, enabling  $n$ -best outputs and compatibility with disjunctive inputs
- **Results:** Experiments with OpenCCG indicate that glue rules yield substantially improved realizations in comparison to greedy fragment assembly

## Motivation

- **Robustness continues to be a problem for chart realizers:** since Kay's (1996) pioneering work, broad coverage realizers have been developed for HPSG, LFG and CCG, but none come close to 100% coverage (see paper)
- With realization shared tasks, “non-native” inputs make robustness even more of an issue
- **By contrast, recent statistical dependency approaches** (Guo et al., 2008, 2010; Gali and Venkatapathy, 2009; Bohnet et al., 2010)—which eschew explicit grammatical constraints—**easily achieve 100% coverage**
- **Here:** use MT-inspired **glue rules** (Chiang, 2007) as a fall-back option, **emulating dependency realization** in cases where no grammatically complete realization can be found
- **Implementation:** Approach formalized in Combinatory Categorical Grammar (Steedman, 2000) and implemented in OpenCCG, improving upon earlier greedy fragment concatenation method; should be applicable to other grammatical frameworks as well

## Glue Rules

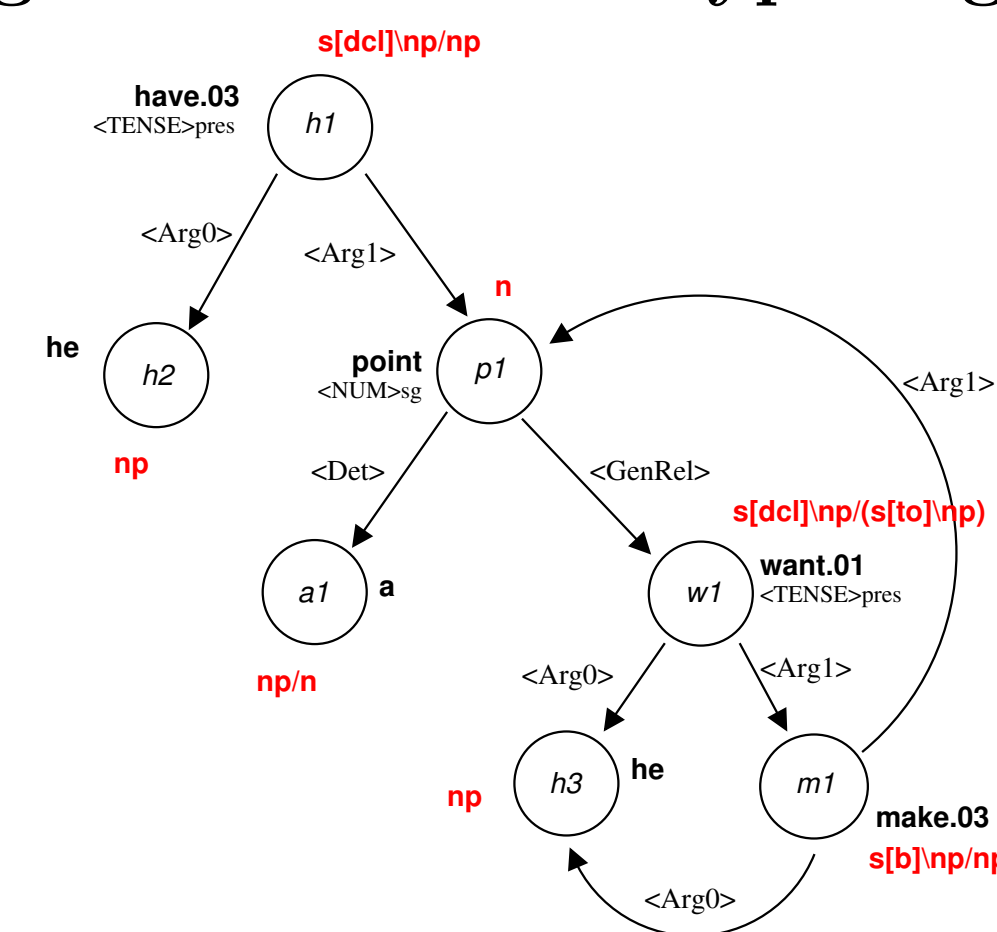
- **Basic Idea:** Concatenate grammatical constituents, avoiding spurious ambiguity

$$\frac{X \ Y[-\text{frag}] \ G}{\text{frag}}$$

- **A Twist:** Glue rules invoked only after chart completed with no complete realization, and limited to empty cells; agenda sorted by edge coverage then model score (i.e. preference given to larger, grammatical edges)
- **Another:** Relaxing the LF chunking constraints (see White, 2006), missing elementary predications (EPs) are made optional, as are ones associated with instantiated unary rules (see paper for related work on grammar-based error detection)
- **A Third:** To allow glue rules to be applied recursively, fragments that complete an LF chunk or disjunction are marked as completed fragments (**frag<sub>c</sub>**), so that they may be used with the glue rule as the right category (where fragments are normally disallowed)
- **Emulating Dependency Realization:** Since LF chunking constraints are applied as usual, the fragment gluing phase becomes tantamount to exploring different permutations of heads and phrases headed by their dependents, much as in dependency realization; 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 to search constraints), with preference given to the orderings with the best model scores

## CCG Surface Realization

### Logical Forms and Hypertagging



### Chart Realization

<i>He</i>	<i>has</i>	<i>a</i>	<i>point</i>	<i>he</i>	<i>wants</i>	<i>to</i>	<i>make</i>
np	s <sub>del</sub> \np\np	np/n	n	np	s <sub>del</sub> \np/(s <sub>to</sub> \np)	s <sub>to</sub> \np/(s <sub>b</sub> \np)	s <sub>b</sub> \np\np
		np	s/(s\np)			s <sub>to</sub> \np\np	
						s <sub>del</sub> \np\np	
						s <sub>del</sub> \np	
						np\np	
						np	
						s <sub>del</sub> \np	
						s <sub>del</sub>	

- Logical forms leave out function words such as relative pronouns and infinitival-*to*
- Realizations are ranked using language models and an averaged perceptron model

## Basic Dependency Ordering Model

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}

- Features count the occurrences of head-dependent and sibling dependent ordering configurations
- No use of CCG categories, so should work even with glue rules

## Evaluation

- A lexico-grammar extracted from CCGbank Sections 02–21 was used to realize the LFs
- With fragmentary realizations, glue rules were compared to the earlier greedy assembly method
- Averaged perceptron models were used for scoring, with and without the dependency features (note that named entity and agreement features were not used in this work)
- An oracle model using an n-gram precision score (approximating BLEU) provided a topline

## Conclusions and Future Work

- Glue rules enhance robustness by providing a fall-back option when no grammatically complete realization can be found
- Unlike an earlier technique of greedily assembling fragments, glue rules enable  $n$ -best outputs, are compatible with disjunctive inputs, and explore a larger search space
- They 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 glue rules can yield improved realizations, though a sizeable gap in quality remains between grammatically complete and fragmentary realizations
- In future work, we plan to experiment with realization ranking models incorporating richer dependency-based features, and to examine the impact of such models on shared task results

## Acknowledgements

This work was supported in part by NSF grant IIS-0812297. Thanks to Chris Brew for helpful discussion.

## Glue Rules Example

### Syntactic Derivation

<i>continue</i>	<i>through</i>	<i>four</i>	<i>(traffic)</i>	<i>lights</i>
s <sub>b</sub> \np	s\s\np	n/n	∅	n
	s <sub>imp</sub>		n	
				OC
				G
				frag <sub>c</sub>

Derivation 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 (see paper)

### Broken HLDS Input

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}

## Realization Results

### Development Set

CCGbank Sect. 00 (1575 complete, 322 fragmentary)

System	BLEU		oracle
	perceptron -deps	perceptron +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

### Test Set

CCGbank Sect. 23 (1932 complete, 328 fragmentary)

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

## Discussion

- On the fragmentary cases, the **glue rules yield more than a 3.5 BLEU point improvement** over greedy fragment assembly; nevertheless the gap between fragmentary and complete items remains large
- The devset improvement is similar both with and without the dependency features
- 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