Mapping between English Strings and Reentrant Semantic Graphs

Fabienne Braune[1], **Daniel Bauer**[2], **Kevin Knight**[3]

[1]Universität Stuttgart, [2]Columbia University, [3]University of Southern California

Abstract

We investigate formalisms for capturing the relation between semantic graphs and English strings. Semantic graph corpora have spurred recent interest in graph transduction formalisms, but it is not yet clear whether such formalisms are a good fit for natural language data—in particular, for describing how semantic reentrancies correspond to English pronouns, zero pronouns, reflexives, passives, nominalizations, etc. We introduce a data set that focuses on these problems, we build grammars to capture the graph/string relation in this data, and we evaluate those grammars for conciseness and accuracy.

Keywords: Corpus (creation), Grammar and Syntax, Semantics

1. Introduction

String transduction via finite-state devices has proven valuable in numerous applications and tree transducers have been successfully applied to machine translation (Knight and Graehl, 2005). As we move to deeper natural language understanding (NLU) and generation (NLG), we encounter semantic graphs. In these graphs, a single entity often plays multiple semantic roles, so the node representing that entity has multiple parents (graph reentrancy). Several corpora containing this type of data have become available (Oepen et al., 2002; King et al., 2003; Basile et al., 2012; Banarescu et al., 2013).

Semantic graphs have spurred interest in graph transduction formalisms that capture the mapping between semantic graphs and natural-language strings (Moore, 1989; Pollard and Sag, 1994; Kroeger, 2004; Bohnet and Wanner, 2001; Jones et al., 2012). If these formalisms have nice computational properties—invertibility, trainability, efficient transduction, etc.—then we may reap the same rewards found in the worlds of string and tree transduction.

However, a formalism must also be a good fit for the data in the sense that a human expert must be able to concisely devise a grammar for this data. In this paper, we start with a set of reentrant semantic graphs paired with English sentences. We ask whether different computationally-attractive formalisms can concisely capture this set of graph/string pairs. Our data is drawn from a microworld, but its graph/string mapping is challenging, as semantic reentrancies are expressed in complex ways by English pronouns, zero pronouns, reflexives, nominalizations, passives, etc.

Our approach is to build grammars to capture the graph/string mapping, apply those grammars in both directions, and measure their accuracy and conciseness in terms of the number of rules required by expert grammar engineers. We only consider computationally attractive formalisms and investigate their ability to elegantly capture the studied data set. Approaches such as (Pollard and Sag, 1994) or (Kroeger, 2004) that do not offer efficient transduction in terms of worst case time complexity, are not discussed here. In addition, we release our data set with the hope that others can improve on these results by identifying formalisms that better fit the data, while still admitting efficient inference. As good formal models are identified, and as larger semantic corpora become available, we will be in a good position to design automatic rule extraction algorithms for fueling NLU and NLG systems.

2. Data

Our data consists of 10,000 semantic graphs over the predicates WANT and BELIEVE, entities BOY and GIRL, and relations ARG0 and ARG1. Each graph comes with up to 10 English realizations.

Sample graphs are shown in Figure 1. A graph may contain any number of predicate instances, but at most one entity of each gender. Each predicate instance may have zero, one, or two arguments, and a single entity may play multiple roles in multiple predicates.

The examples in Figure 1 demonstrate various English mechanisms, including passives ("is believed") and control structures ("wants the girl to believe"). The data also includes nominalizations ("her desire for him"), reflexives ("wants herself"), long-distance dependencies ("she wants to want to . . . want herself" rather than "*she wants to want to . . . want her"), and other phenomena.

We produce our data by writing two programs, one that systematically enumerates all semantic graphs in this domain (starting from the smallest), and another that generates English from them. Thus, we can create as much data as we would like. The English generation program is arbitrary, i.e., not written in a declarative rule formalism. We make our data publicly available at http://amr.isi.edu/download/boygirl.tgz

3. Formalisms

We seek to capture this data concisely in an efficiently-processable formalism. In particular, the formalism should be invertible, and it should support efficient forward and backward application (graph-to-string and string-to-graph), efficient k-best generation, and efficient EM weight training.

We devise and test three solutions:

1. Our first solution uses a single synchronous hyperedge-replacement grammar (SHRG) (Jones et al., 2012). A SHRG captures a graph/string relation and may be used to map in either direction.

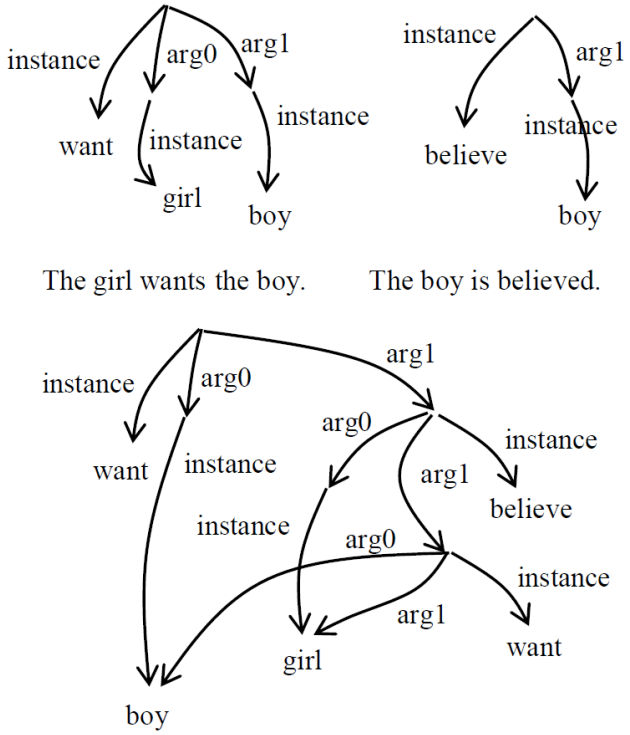

Figure 1: Sample semantic graphs and English realizations.

2. Our second solution is a cascade of two devices: a bottom-up DAG-to-tree transducer (D2T) (Kamimura and Slutzki, 1982), followed by a top-down tree-to-string transducer (LNTs) (Comon et al., 1997). The latter's job is to take the yield of trees produced by the D2T.

3. Our third solution is also a cascade, separating different linguistic processes into different machines. First, a D2T tree-ifies the graph. Then two extended tree-to-tree transducers (xLNT) (Maletti et al., 2008) respectively introduce verbs and pronouns. Finally, an LNTs converts the produced trees into strings.

Because we have software that implements generic operations on these types of automata (May and Knight, 2006; Chiang et al., 2013; Quernheim and Knight, 2012), we can readily test any manually-engineered solution on our 10,000 data cases, in both graph-to-string and string-to-graph directions.

Our research seeks to determine how easy it is to explain the data with various solutions, i.e., to find out how well they fit the natural language data. It is not concerned with their purely theoretical properties, which are well-described elsewhere.

3.1. Synchronous hyperedge-replacement grammar

Hyperedge Replacement Grammars (HRG, **?**)) describe languages of graphs (for instance, to capture all graphs in figure 1) similar to how context free string grammars (CFG) describe string languages. The bottom part of figure 2 illustrates how HRG constructs graphs. We begin with the start symbol S and apply rule R1 to replace it with an initial reentrant graph that contains further nonterminal edges (EF and SF). After this, nonterminal edges are replaced recursively by splicing in graph fragments in a manner reminiscient to tree-adjoining grammar (TAG) (Joshi and Schabes, 1997). A graph fragment is inserted in place of a nonterminal edge by fusing certain nodes of the fragment with the nodes connected by the edge.

Therefore, two semantic nodes that point to the same entity (e.g. the root note of the graph and the ARG1 of that node point to the ARG0 of the root node) may be initially close together, but then get pushed apart in the derivation. This mechanism allows us to remember referents that need to be referred back to later in the derivation. In HRG, nonterminal edges can be hyperedges, which are edges that connect an arbitrary number of nodes. Hyperedges can keep track of multiple reentrant nodes at the same time, so that graph fragments fused in later in the derivation can access them.

To translated between english strings and semantic graphs we use synchronous HRG (SHRG, Jones et al. (2012)), which pairs every HRG rule with a context free string grammar (CFG) rule by synchronizing non-terminals. The first rule in figure 2 introduces a graph that can be glossed "EF believes *some SF about herself"* and pairs it with the string "EF believes SF". The SF nonterminal is replaced by rule R2, which itself introduces another non-terminal edge (labeled Sinf). R3 continues the derivation, introducing a second WANT predicate in the semantics and a zero pronoun (0) in the English string. This zero pronoun and the corresponding graph edge (ARG0) realizes the reference to the EF entity we kept track of during the derivation.

SHRG also permits horizontal construction of graphs, which lets us incrementally produce highly reentrant graph nodes. Most other grammar and automata formalisms (including context-free grammars) permit only vertical construction, creating or consuming all of a node's edges at once.

The biggest weakness in using SHRG in this way is that we still use CFG for the language of English strings, especially since we require our grammars to be reversible. It is not sufficient to write an over-generating grammar that can be used for language understanding, since the same grammar has to be used for generation as well. With plain CFG there is no easy way to make sure pronominal forms are compatible with the verb and pronoun gender agrees over long distances. The features of all entities that interact over long distances need to be encoded in the nonterminal alphabet, quickly leading to combinatorial explosion of the grammar. For instance, the grammar used in figure 2 contains a nonterminal symbol SF to indicate that any entity edge labeled with SF points to has female gender. Similarly the nonterminal Sinf indicates a subphrase with a main verb in infinitive form (required by the control construction), that points back to a female entity.

3.2. DAG-to-tree transducer

The D2T transducer operates bottom-up on an input graph. After consuming a semantic node by rule, it outputs an English tree fragment along with state information. English

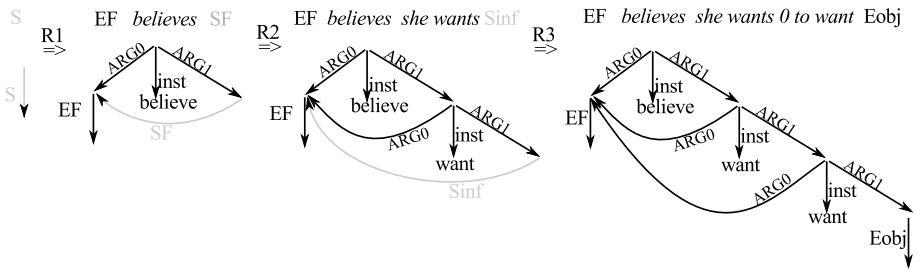

Figure 2: A partial SHRG derivation that simultaneously builds a string (top) and graph (bottom).

tree fragments can be combined by subsequent rules that match the states of those fragments.

We initially use D2T rules to split reentrant graph nodes. In Figure 3, we split the GIRL node into three English fragments with states q_{ps}, q_{ps}, and q_{pz}. The rules used in this derivation step are shown in Figure 6. Via other derivations, different substitutions and state assignments are pursued non-deterministically. Subsequent rules observe predicates higher in the graph, and they confirm that the states can be used to express those predicates. Figure 7 shows rules used in subsequent steps of the derivation in Figure 3.

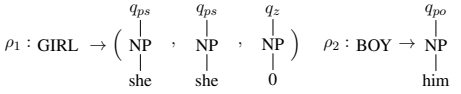

Figure 6: D2T rules in first derivation step

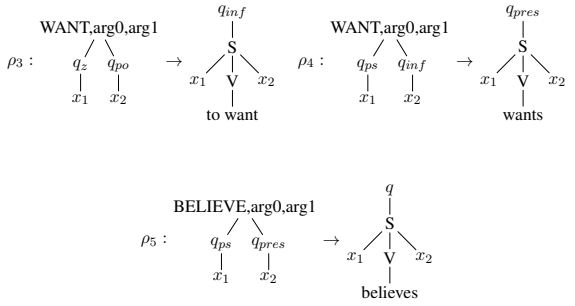

Figure 7: D2T rules in subsequent derivation steps

The D2T transducer allows us to elegantly model the mapping between graph reentrancies and agreement of pronominal genre over arbitrarily long distances. In the graph-string pairs shown in Figures 4 and 5, the GIRL node has two reentrancies realized by the pronouns "she" and "her". In the first pair, these reentrancies are separated by a single instance of the predicate WANT and in the second pair, by two such instances. To generate the associated English strings, we split, in *both* derivations, the GIRL node into two English fragments with states q_{ps}, q_{po} and leaves "she","her". This means that in both cases we use the same D2T rule, shown in Figure 4. The same rule can again be used to process all graph-string pairs where the node GIRL has two reentrancies realized by the pronouns "she" and "her" regardless of the number of nodes that separate these reentrancies. On the other hand, D2T rules can only be reused when reentrancies are realized by the same pronominal forms. Large numbers of reentrancies realized by diverse surface forms are hard to capture using this device. This weakness is due to the fact that when D2T consumes a node in the input DAG, all reentrancies have to be processed in one step. As an illustration, we replace the first WANT predicate in Figure 3 with the predicate BELIEVE. The resulting graph-string pair is displayed in Figure 8. In the corresponding derivation, rule ρ_1, which we used in Figure 3, cannot be used anymore to split the GIRL node because the third reentrancy into this node is now realized by a different pronoun. Hence, a new rule ($\rho 6$ in Figure 8) has to be written to process our new graph-string pair. Such new rules have to be created each time a reentrancy is realized by a different surface form. For large numbers of reentrancies this quickly leads to an explosion in the number of rules.

3.3. Cascade of tree transducers

Complex linguistic processes are often most elegantly modeled with a sequence of simple devices (Pereira et al., 1994; Knight and Graehl, 1998; Knight and Al-Onaizan, 1998). The model effectively makes several passes over the structure, with each pass implementing a targeted transformation. Here, we first convert the graph into a tree using a D2T transducer. We specify abstract entity labels instead of pronominal and verbal forms in the leaves of our D2T rules. Then we employ an xLNT transducer to introduce verbal forms such as finite verbs and infinitives. A second xLNT models the relation between entity nodes and pronouns. Figure 9 illustrates the input graph in Figure 3 processed using our cascade of devices. The rules of the D2T transducer in the cascade are shown in Figure 10.

The cascade of transducers allows us to capture our data set in a more concise way than a single SHRG grammar or D2T device. Its main adavantage is that it breaks down the complex graph to string mapping given in our data into simple tasks. Using rules that specify abstract labels instead of surface forms in the initial D2T allows us to keep this device small. In particular, we avoid the combinatorial explosion of rules necessary to capture reentrancies realized by different surface forms. Similarly, the fact that we model verbal

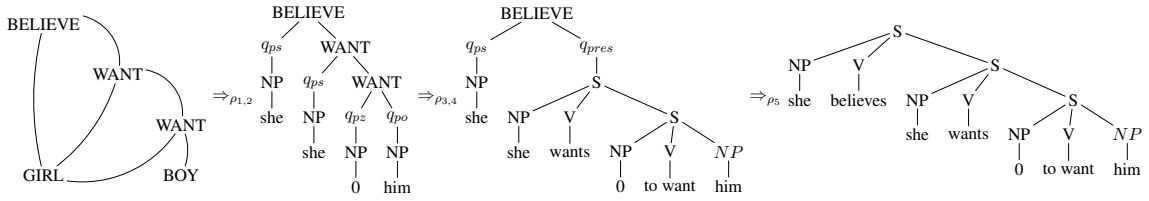

Figure 3: A sample D2T derivation.

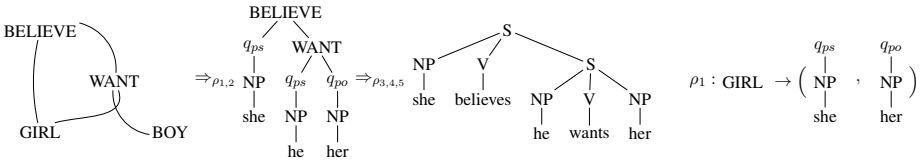

Figure 4: Pronominal agreement over short distance

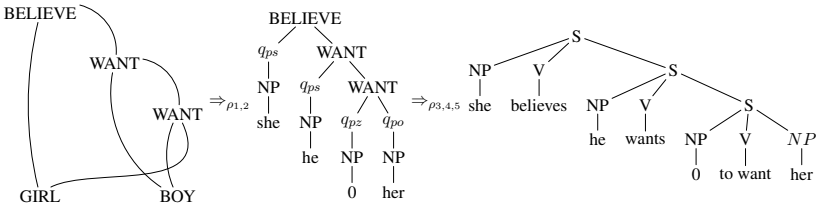

Figure 5: Pronominal agreement over longer distance

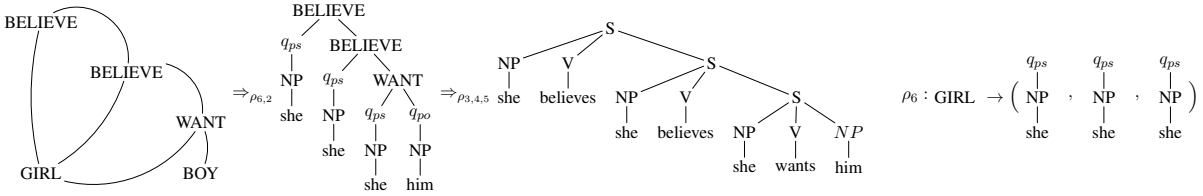

Figure 8: Reentrancies with various surface forms

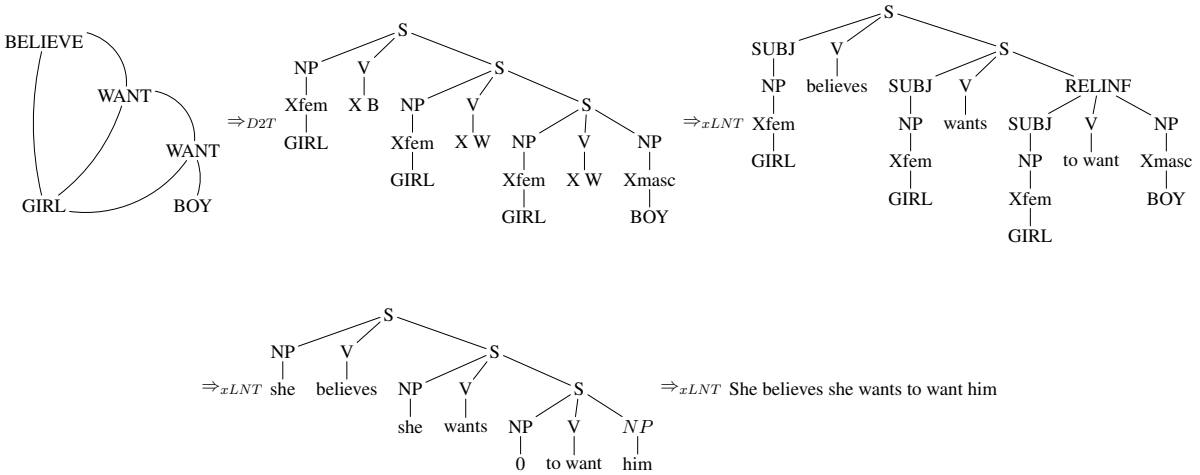

Figure 9: Semantic graph processed using D2T followed by a cascade of XTOPs.

tenses and pronominal forms using two xLNT allows us to keep these devices small.

4. Empirical Evaluation

In our empirical evaluation, we measure how accurately and concisely each discussed formalism allows us to model

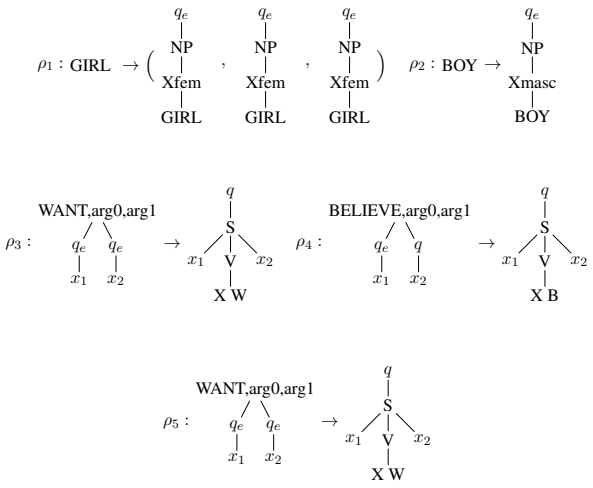

Figure 10: Rules of initial D2T in cascade

our data. For generation of strings from graphs (natural language generation), we measure coverage as well as accuracy of the produced strings. To this aim, we apply each grammar or device to generate a string for each of the 10,000 input graphs. Coverage indicates whether any output was generated at all. Accuracy is measured against our 10 English references using BLEU (Papineni et al., 2002). For generation of graphs from strings (natural language understanding), we apply each device backwards and generate a graph from each of the 98,818 strings in the data set. Again, we measure coverage as well as accuracy. For accuracy, we use Smatch (Cai and Knight, 2013). For both tasks, we measure conciseness by counting the number of rules in each solution, and the number of bytes.

Figure 11 shows the results. The strings obtained in the generation task achieve full coverage and reasonable good accuracy. The cascading approach requires the fewest number of rules. This is mainly due to the capacity of cascades to cut down complex tasks into simpler ones. The DAG-to-tree transducer requires the largest number of rules. This is mainly due to its inability to deal with long co-reference chains. In between, the amount of SHRG rules is due to the necessity to encode agreement information in non-terminal edges. Note that SHRG is somewhat less accurate than the other approaches because it permits more flexibility in generating co-reference chains. The graphs obtained in the understanding task achieve low coverage, meaning that none of the devices is able to fully process the 10,000 first reference strings. For all solutions, attempting to increase coverage leads to a large increase in the number of rules. However, for the generated graphs accuracy is reasonably good. As a conclusion, we can say that although our data set confirms the observations in Section 3, none of the investigated formalisms elegantly captures the full complexity of our data set, especially for the understanding task. Our data is publicly available, and we hope other computationally attractive formalisms can be shown to capture the relation between English strings and reentrant graphs more concisely and accurately.

5. Conclusion

We created a corpus of highly reentrant semantic graphs associated to English strings. We presented several approaches to capture the graph/string relation found in this data set. The large number of rules required by these formalisms to model this mapping shows that there is a lot of room for improvement. We release our data set in the hope that others will download it and devise more elegant formalisms to capture the relation it contains.

6. References

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. (2013). Abstract meaning representation for sembanking. In *Proc. ACL Linguistic Annotation Workshop*.

Basile, V., Bos, J., Evang, K., and Venhuizen, N. (2012). Developing a large semantically annotated corpus. In *Proc. LREC*.

Bohnet, B. and Wanner, L. (2001). On using a parallel graph rewriting formalism in generation. In *Proc. EWNLG Workshop*.

Cai, S. and Knight, K. (2013). Smatch: an evaluation metric for semantic feature structures. In *Proc. ACL*.

Chiang, D., Andreas, J., Bauer, D., Hermann, K. M., Jones, B., and Knight, K. (2013). Parsing graphs with hyperedge replacement grammars. In *Proc. ACL*.

Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., and Tommasi, M. (1997). Tree automata techniques and applications. Available on `www.grappa.univ-lille3.fr/tata`. release October, 1st 2002.

Jones, B., Andreas*, J., Bauer*, D., Hermann*, K. M., and Knight, K. (2012). Semantics-based machine translation with hyperedge replacement grammars. In *Proc. COLING*. *first authorship shared.

Joshi, A. and Schabes, Y. (1997). Tree-adjoining grammars. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages (Vol. 3)*. Springer.

Proposed Solution	# of rules	# of bytes	NLG coverage	NLG accuracy (Bleu)	NLU coverage	NLU accuracy (Smatch)
SHRG	242	18035	100	88.13	4.9	90
D2T	361	21533	100	90.13	3.9	83
Cascade	105	3843	100	90.13	4.3	79

Figure 11: Results of three automata solutions applied to our graph/string data.

Kamimura, T. and Slutzki, G. (1982). Transductions of dags and trees. *Theory of Computing Systems*, 15(1).

King, T. Holloway, Crouch, R., Riezler, S., Dalrymple, M., and Kaplan, R.M. (2003). The PARC 700 dependency bank. In *Proc. LINC Workshop*.

Knight, K. and Al-Onaizan, Y. (1998). Translation with finite-state devices. In *Proc. AMTA*.

Knight, K. and Graehl, J. (1998). Machine transliteration. *Computational Linguistics*, 24(4).

Knight, K. and Graehl, J. (2005). An overview of probabilistic tree transducers for natural language processing. In *Proc. CICLING*.

Kroeger, Paul. (2004). *Analyzing Syntax : A Lexical-Functional Approach*. Cambridge University Press.

Maletti, A., Graehl, J., Hopkins, M., and Knight, K. (2008). The power of extended top-down tree transducers. *SIAM J. Computing*, 39(2).

May, J. and Knight, K. (2006). Tiburon: A weighted tree automata toolkit. In *Proc. CIAA*.

Moore, R. C. (1989). Unification-based semantic interpretation. In *Proc. ACL*.

Oepen, S., Toutanova, K., Shieber, S., Manning, C., Flickinger, D., and Brants, T. (2002). The LinGO redwoods treebank : Motivation and preliminary applications. In *Proc COLING*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proc. ACL*.

Pereira, F., Riley, M., and Sproat, R. (1994). Weighted rational transductions and their application to human language processing. In *Proc. HLT*.

Pollard, Carl J. and Sag, Ivan A. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Quernheim, D. and Knight, K. (2012). DAGGER: a toolkit for automtata on directed acyclic graphs. In *Proc. FSMNLP*.