Hyperedge Replacement and Nonprojective Dependency Structures

Daniel Bauer and **Owen Rambow**
Columbia University
New York, NY 10027, USA
{bauer,rambow}@cs.columbia.edu

Abstract

Synchronous Hyperedge Replacement Graph Grammars (SHRG) can be used to translate between strings and graphs. In this paper, we study the capacity of these grammars to create non-projective dependency graphs. As an example, we use languages that contain cross serial dependencies. Lexicalized hyperedge replacement grammars can derive string languages (as path graphs) that contain an arbitrary number of these dependencies so that their derivation trees reflect the correct dependency graphs. We find that, in contrast, string-to-graph SHRG that derive dependency structures on the graph side are limited to derivations permitted by the string side. We show that, as a result, string-to-graph SHRG cannot capture languages with an unlimited degree of crossing dependencies. This observation has practical implications for the use of SHRG in semantic parsing.

1 Introduction

Hyperedge Replacement Grammars (HRG) are a type of context free graph grammar. Their derived objects are hypergraphs instead of strings. A synchronous extension, Synchronous Hyperedge Replacement Grammars (SHRG) can be used to translate between strings and graphs. To construct a graph for a sentence, one simply parses the input using the string side of the grammar and then interprets the derivations with the graph side to assemble a derived graph.

SHRG has recently drawn attention in Natural Language Processing as a tool for semantic construction. For example, Jones et al. (2012) propose to use SHRG for semantics based machine translation, and Peng et al. (2015) describe an approach to learning SHRG rules that translate sentences into Abstract Meaning Representation (Banarescu et al., 2013).

Not much work has been done, however, on understanding the limits of syntactic and semantic structures that can be modeled using HRG and SHRG. In this paper, we examine syntactic dependency structures generated by these formalisms, specifially whether they can create correct dependency trees for non-projective phenomena. We focus on non-projectivity caused by copy language like constructions, specifically cross-serial dependencies in Dutch. Figure 1 shows a (classical) example sentence containing such dependencies and a dependency graph.

This paper looks at dependency structures from two perspectives. We first review HRGs that derive string languages as path graphs. The set of these languages is known to be the same as the languages generated by linear context free rewriting systems (Weir, 1992). We consider HRG grammars of this type that are lexicalized (each rule contains exactly one terminal edge), so we can view their *derivation trees* as dependency structures. We provide an example string-generating HRG that can analyze the sentence in Figure 1 with the correct dependency structure and can generate strings with an unlimited number of crossing dependencies of the same type.

Under the second perspective, we view the *derived graphs* of synchronous string-to-HRG grammars as dependency structures. These grammars can generate labeled dependency graphs in a more flexible way, including labeled dependency edges, local reordering of dependencies (allowing a more semantically oriented analysis of prepositional phrases and conjunctions), structures with arbitrary node degree, and reentrancies. We present a grammar to analyze the string/graph pair in Fig-

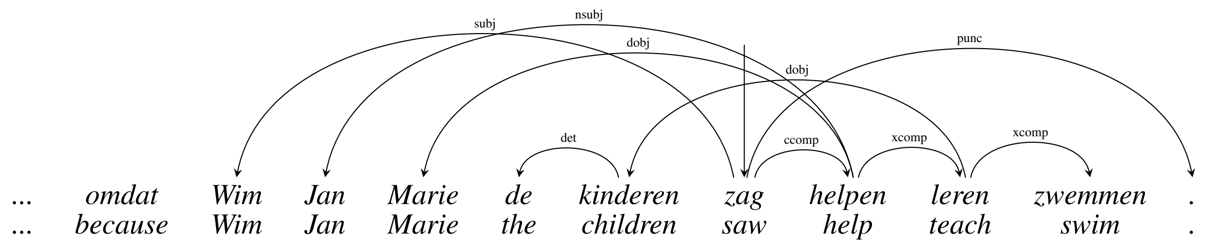

subj
nsubj
dobj
dobj
punc
det
ccomp
xcomp
xcomp
... omdat Wim Jan Marie de kinderen zag helpen leren zwemmen .
... because Wim Jan Marie the children saw help teach swim .

Figure 1: Example sentence illustrating cross-serial dependencies in Dutch. English translation: "*because Wim saw Jan help Marie teach the children to swim.*"

ure 1, that derives the correct labeled dependency structure, but whose derivation does not resemble syntactic dependencies. Using this example, we observe an important limitation of string-to-graph SHRG: With nonterminal hyperedges of bounded type (number of incident vertices), we cannot analyze cross-serial dependencies with an unlimited number of crossing edges. Specifically, for a given dependency edge covering a span of words, the number of nodes outside the span that can have a dependent or parent inside the span is limited. This is because, on the input side, the grammar is a plain string CFG. In a string CFG derivation, each node must correspond to a connected subspan of the input. Because of this constraint on the derivation, the dependency subgraphs constructed by the HRG must maintain a reference to all words that have a long distance dependent elsewhere in the string. These references are passed on through the derivation in the external nodes of each graph rhs of the SHRG rules. External nodes are special vertices at which graph fragments are connected to the surrounding graph.

To avoid this problem, instead of a plain string CFG one can use other formalisms that produce context free derivation trees, such as the string-generating HRGs we discuss in this paper or LTAG.

Semantic representations, such as Abstract Meaning Representation, resemble dependency structures. Therefore, while we do not discuss semantic graphs to skirt the issue of reentrancy, non-projective linguistic phenomena that appear in syntactic dependency structure are also relevant when translating strings into semantic representations. We believe that our observations are not only of theoretical interest, but affect practical applications of SHRG in semantic parsing.

The paper proceeds as follows: Section 2 provides a formalization of Hyperedge Replacement Grammars and introduces necessary terminology. In section 3, we discuss string generating HRGs and illustrate how they can be used to correctly analyze cross-serial dependencies in an example. Section 4 examines string-to-graph SHRGs and observes their limitations in generating cross-serial dependencies. In section 5, we analyze this limitation in more detail, demonstrating a relationship between the *order* of a grammar (the maximum hyperedge type) and the maximum number of edges crossing another edge. Section 6 provides an overview of related work. Finally, we conclude and summarize our findings in section 7.

2 Hyperedge Replacement Graph Grammars

A *directed, edge-labeled hypergraph* is a tuple $H = \langle V, E, \ell \rangle$, where V is a finite set of vertices, $E \subseteq V^+$ is a finite set of hyperedges, each of which connects a number of vertices, and ℓ is a labeling function with domain E. The number of vertices connected by a hyperedge is called its *type*.

A *hyperedge replacement grammar* (HRG, Drewes et al. (1997)) is a tuple $\mathcal{G} = \langle N, \Sigma, P, S \rangle$ where N is a ranked, finite set of nonterminal labels, Σ is a finite set of terminal labels such that $\Sigma \cap N = \emptyset$, $S \in N$ is the designated start symbol, and P is a finite set of rules. Each rule $r \in P$ is of the form $(A \to R, X)$, where $A \in N$, $R = \langle V, E, \ell \rangle$ is a hypergraph with $\ell : E \to N \cup T$, and $X \in V^*$ is a list of *external nodes*. We call the number of vertices $|V|$ in a rule rhs the *width* of the rule. The maximum type of any nonterminal hyperedge in the grammar is called the *order* of the grammar.[1]

[1]We choose the term *order* instead of *rank*. Both terms

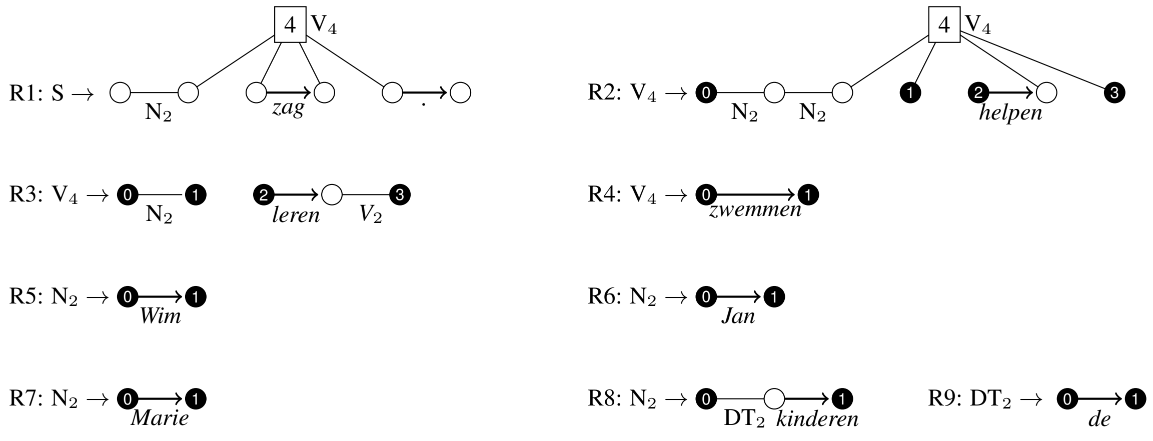


Figure 2: A 'string-generating' lexicalized hyperedge replacement grammar for Dutch cross serial dependencies. The grammar can derive the sentence in figure 1. The derivation tree for this sentence represents the correct dependency structure.

Given a partially derived graph H we can use a rule $(A \rightarrow R, X)$ to rewrite a hyperedge $e = (v_1, \cdots, v_k)$ if e has label A and $k = \mathit{length}(X)$. In this operation, e is removed from H, a copy of R is inserted into H and the external nodes $X = (u_1, \cdots, u_k)$ of the copy of R are *fused* with the nodes connected by e, such that u_i is identified with v_i for $i = 1, \ldots, k$.

When showing rules in diagrams, such as Figure 2, we draw external nodes as black circles and number them with an index to make their order explicit. Nonterminal hyperedges are drawn as undirected edges whose incident vertices are ordered left-to-right.

The relation $H \Rightarrow_{\mathcal{G}} H'$ holds if hypergraph H' can be derived from hypergraph H in a single step using the rules in $\mathcal{G}$. Similarly $H \Rightarrow^*_{\mathcal{G}} H'$ holds if H' can be derived from H in a finite number of steps. The *hypergraph language* of a grammar $\mathcal{G}$ is the (possibly infinite) set of hypergraphs that can be derived from the start symbol S. $L(\mathcal{G}) =$

$$\bigcup_{(S \rightarrow H, \langle\rangle) \in P} \{H \Rightarrow^*_{\mathcal{G}} H' | H' \text{ has only terminals }\}$$

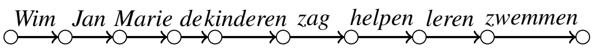
We will show examples for HRG derivations below.

HRG derivations are context-free in the sense that the applicability of each production depends on the nonterminal label and type of the replaced edge only. We can therefore represent derivations as trees, as for other context free formalisms. Context freeness also allows us to extend the formalism to a synchronous formalism, for example to translate strings into trees, as we do in section 4. We can view the resulting string and graph languages as two interpretations of the same set of possible derivation trees described by a regular tree grammar (Koller and Kuhlmann, 2011).

are used in the literature. We use the word *rank* to refer to the maximum number of nonterminals in a rule right hand side.

3 HRG Derivations as Dependency Structures

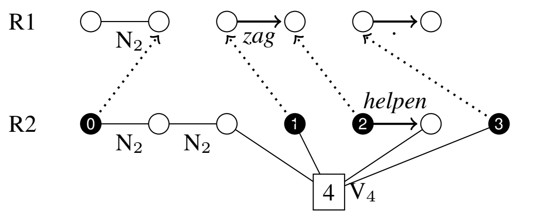
We first discuss the case in which HRG is used to derive a sentence and examine the dependency structure induced by the derivation tree. Hyperedge Replacement Grammars can derive string languages as path graphs in which edges are labeled with tokens. For example, consider the path graph for the sentence in Figure 1.



Engelfriet and Heyker (1991) show that the string languages generated by HRG in this way are equivalent to the output languages of Deterministic Tree Walking Transducers (DTWT). Weir (1992) shows that these languages are equivalent to the languages generated by linear context free rewriting systems (LCFRS) and that the LCFRS languages with fan-out k are the same as the HRG string languages with order $2k$.

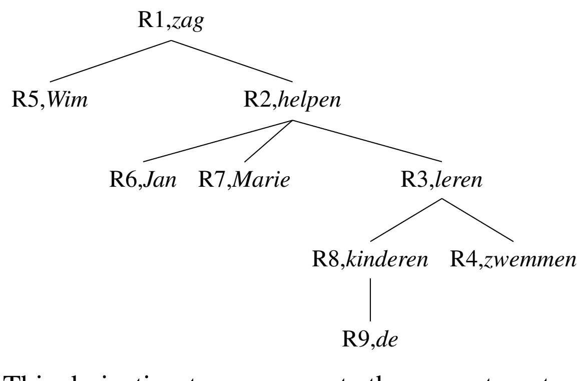
The analysis of cross-serial dependencies has been studied in a number of 'mildly context sensitive' grammar formalisms. For example, Rambow and Joshi (1997) show an analysis in LTAG. Because the string languages generated by these formalisms are equivalent to languages of LCFRS with fan-out 2, we know that we must be able to write an HRG of order 4 that can capture cross-serial dependencies.

Figure 2 shows such a string generating HRG that can derive the example in Figure 1. Each rule rhs consists of one or more internally connected spans of strings paths of labeled edges. The external nodes of each rhs graph mark the beginning and end of each span. The nonterminal labels of other rules specify how these spans are combined and connected to the surrounding string. For illustration, consider the first two steps of a derivation in this grammar. Rule 1 introduces the verb '*zag*' and its subject. Rule 2 inserts '*helpen*' to the right of '*zag*' and its subject and direct object to the right of the subject of '*zag*'. This creates crossing dependencies between the subjects and their predicates in the derivation.



The partially derived graph now contains a span of nouns and a span of verbs. The nonterminal hyperedge labeled V_4 indicates where to append new nouns and where to add new verbs. Note that rule 2 (or an identical rule for a different verb) can be re-applied to generate cross-serial dependencies with an arbitrary number of crossings. It is easy to see that grammars of this type correspond to LCFRS almost directly.

Using the grammar in Figure 2, there is a single derivation tree for the example sentence in Figure 1.



This derivation tree represents the correct syntactic dependency structure for the sentence. This is not the case for all lexicalized 'mildly context sensitive' grammar formalisms, even if it is possible to write grammars for languages that contain cross-serial dependencies. In TAG, long distance dependencies are achieved using adjunction. Both dependents are introduced by the same auxiliary tree, stretching the host tree apart. An LTAG derivation for the example sentence would start with an elementary tree for '*zwemmen*' and then adjoin '*leren*'. The resulting dependency structure is therefore inverted.

4 Deriving Dependency Graphs with Synchronous String-to-Graph Grammars

We now consider grammars whose *derived* graphs represent the dependency structure of a sentence. The goal is to write a synchronous context-free string-to-graph grammar that translates sentences into their dependency graphs. If the string side of the grammar is a plain string CFG, as we assume here, the derivation cannot reflect non-projective dependencies directly. Instead, we must use the graph side of the grammar to assemble a dependency structure.

This approach has several potential advantages in applications. In the string-generating HRG discussed in the previous section, the degree of a node in the dependency structure is limited by the rank of the grammar. Using a graph grammar to derive the graph, we can add an arbitrary number of dependents to a node, even if the rules contributing these dependency edges are nested in the derivation. This is especially important for more semantically inspired representations where all semantic arguments should become direct dependents of a node (for example, deep subjects). We can also make the resulting graphs reentrant. In addition, because HRGs produce labeled graphs, we can add dependency labels. Finally, even though the example grammar in Figure 3 is lexicalized on the string side, lexicalization is no longer required to build a dependency structure. Unfortunately, 'decoupling' the derivation from the dependency structure in this way can be problematic, as we will see.

Figure 3 shows a synchronous hyperedge replacement grammar that can translate the sentence from Figure 1 into its dependency graph. A *synchronous hyperedge replacement grammar* (SHRG) is a synchronous context free grammar in which at least one of the right hand sides uses hypergraph fragments. The two sides of the grammar are synchronized in a strong sense. Both rhs of each grammar rule contain exactly the same instances of nonterminals and the instances

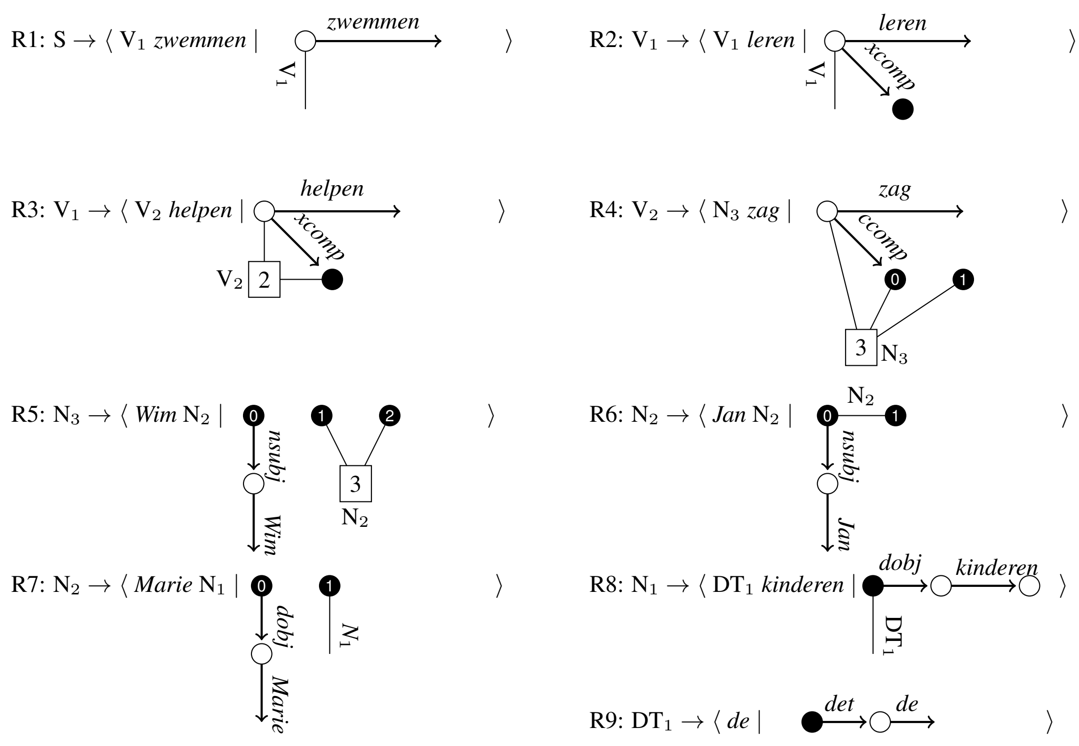
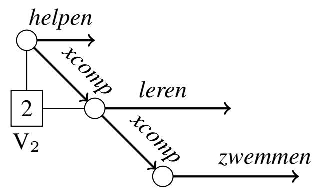


Figure 3: A synchronous string-to-graph grammar for Dutch cross-serial dependencies. The grammar can derive the sentence/dependency graph pair in in Figure 1, but the derivation tree does not reflect syntactic dependencies.

are related by a bijective synchronization relation (in case of ambiguity we make the bijection explicit by indexing nonterminals when representing grammars). In a SHRG, each nonterminal label can only be used to label hyperedges of the same type. For example, V_2 is only used for hyperedges of type 2. As a result, all derivations for the string side of the grammar are also valid derivations for graphs.

In the grammar in Figure 3, vertices represent nodes in the dependency structure (words). Because HRGs derive edge labeled graphs but no vertex labels, we use a unary hyperedge (a hyperedge with one incident vertex) to label each node. For example, the only node in the rhs of rule 1 has the label '*zwemmen*'.

Nonterminal hyperedges are used to 'pass on' vertices that we need to attach a dependent to at a later point in the derivation. External nodes define how these nodes are connected to the surrounding derived graph. To illustrate this, a derivation using the grammar in Figure 3 could start with rule 1, then replace the nonterminal V_1 with the rhs of rule 2. We then substitute the new nonterminal V_1 introduced by rule 2 with rule 3. At this point, the partially derived string is 'V_2 *helpen leren zwemmen*' and the partially derived graph is



.

The nonterminal V_2 passes on a reference to two nodes in the graph, one for '*helpen*' and one for '*leren*'. This allows subsequent rules in the derivation to attach subjects and objects to these nodes, as well as the parent node ('*zag*') to '*helpen*'.

To derive the string/graph pair in Figure 1, the rules of this grammar are simply applied in order (rule 1 ⇒ rule 2 ⇒ ⋯ ⇒ rule 9). Clearly, the resulting derivation is just a chain and bears no resemblance to the syntactic dependency structure.

While the grammar can derive our example sentence, it does not permit us to derive dependency structures with an arbitrary number of crossing dependencies. This is because the nonterminal edges need to keep track of all possible sites at which long distance dependents can be attached at a later point in the derivation. To add more crossing

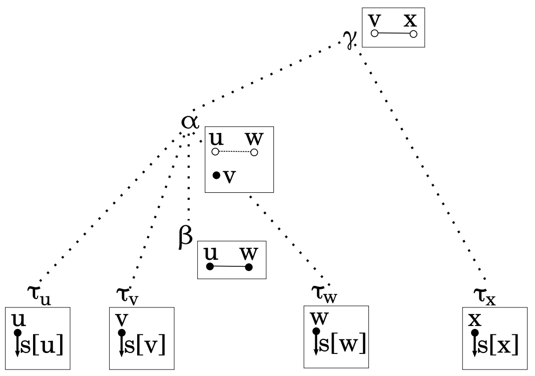


Figure 4: Sketch of the derivation tree of a synchronous hyperedge replacement grammar, showing two dependency edges (u, w) and (v, x), and $u < v < w < x$.The graph fragment associated with the rule at node α needs to contain nodes u, w and v. v must be an external node.

dependencies we therefore need to create special rules with nonterminal hyperedges of a larger type, as well as the corresponding rules with a larger number of external nodes. Because any grammar has a finite number of rules and a fixed order, we cannot use this type of SHRG grammar to model languages that permit an arbitrary degree of crossing edges in a graph. While the graph grammar can keep track of long-distance dependencies, the string grammar is still context free, so any nonlocal information needs to be encoded in the nonterminals. The penalty we pay for being able to remember a limited set of dependents through the derivation is that we need a refined alphabet of nonterminals (V_1, V_2, V_3, $\cdots$; instead of just V).

5 Edge Degree and Hyperedge Type

In section 4 we demonstrate that we need an ever-increasing hyperedge type if we want to model languages in which a dependency edge can be crossed by an arbitrary number of other dependency edges. So far, we have only illustrated this point with an example. In this section we will demonstrate that no such grammar can exist.

It is clear that the problem is not with generating the tree language itself. We could easily extend the string-generating grammar from section 3, whose derivation trees reflect the correct dependency structure, by adding a second graph rhs that derives an image of the derivation tree (potentially with dependency labels). Instead, the problem appears to be that we force grammar rules to be applied according to the string derivation. Specifically, the partially derived string associated with each node in the derivation needs to be a contiguous subspan. This prevents us from assembling dependencies locally.

To make this intuition more formal, we demonstrate that there is a relationship between number of crossing dependencies and the the minimum hyperedge type required in the SHRG. We first look at a single pair of crossing dependency edges and then generalize the argument to multiple edges crossing into the span of an edge. For illustration, we provide a sketch of a SHRG derivation tree in Figure 4.

Assume we are given a sentence $s = (w_0, w_1, \cdots, w_{n-1})$, and a corresponding dependency graph $G = \langle V, E, \ell \rangle$ where $V = \{0, 1, \cdots, n-1\}$. We define the range of a dependency edge (u, v) to be the interval $[u, v]$ if $v > u$ or else $[v, u]$. For each dependency edge (u, v) the number of crossing dependencies is the number of dependency nodes properly outside its range, that share a dependency edge with any node properly inside its range. The degree of crossing dependencies of a dependency graph is the maximum number of crossing dependencies for any of its edges.

Given a SHRG derivation tree for s and G, each terminal dependency edge $(u, w) \in E$ must be produced by the rule associated with some derivation node β (see Figure 4). Without loss of generality, assume that $u < w$. String token $s[u]$ is produced by the rule associated with some derivation node τ_u and $s[w]$ is produced by the rule of some derivation node τ_w. On the graph side, τ_u and τ_w must contain the nodes u and w because they generate the unary hyperedges labeling these vertices. There must be some common ancestor α of β, τ_u, and τ_w that contains both u and w. u and w must be connected in α by a nonterminal hyperedge, because otherwise there would be no way to generate the terminal edge (u, w) in β (note that it is possible that α and β are the same node in which case the rule of this node does not contain a nonterminal edge).

Now consider another pair of nodes v and x such that $u < v < w < x$ and there is a dependency edge $(v, x) \in E$ or $(x, v) \in E$. $s[v]$ is generated by τ_v and $s[x]$ is generated by τ_x. As before, there must be a common ancestor γ of τ_v and τ_x, in which v and x are connected by a nonterminal hyperedge. Because $u < v < w < x$ either

α is an ancestor of γ or γ is an ancestor of α. For illustration, we assume the second case. The case where α dominates γ is analogous.

Since the graph fragments of all derivation nodes on the path from γ to τ_v must contain a vertex that maps to v, α must contain such a vertex. This vertex needs to be an external node of the rule attached to α because otherwise v could not be introduced by γ.

We can extend the argument to an arbitrary number of crossing dependency edges. As before, let (u, w) be a dependency edge and α be the derivation node whose graph fragment first introduces the nonterminal edge between u and w. For *all* dependency edges (x, y) or (y, x) for which y is in the range of (u, w) and x is outside of the range of (u, w) (either $x < u < y < w$ or $u < y < w < x$) there must be some path in the derivation tree that leads through α. All graph fragments on this path contain a vertex mapped to y. As a result, the graph fragment in α needs to contain one external node for each x that has a dependency edge to some node y inside the range (u, w). In other words, α needs to contain as many external nodes as there are nodes outside the range (u, w) that share a dependency edge with a node inside the range (u, w).

Because every HRG has a fixed order (the maximum type of any nonterminal hyperedge), no SHRG that generates languages with an arbitrary number of cross-serial dependencies can exist. It is known that the hypergraph languages $\mathcal{HRL}_k$ that can be generated by HRGs of order k form an infinite hierarchy, i.e. $\mathcal{HRL}_1 \subsetneq \mathcal{HRL}_2 \subsetneq \cdots$ (Drewes et al., 1997). Therefore, the string-to-graph grammars required to generate cross-serial dependencies up to edge degree k are strictly more expressive than those that can only generate edge degree $k - 1$.

6 Related Work

While the theory of graph grammars dates back to the 70s (Nagl, 1979; Drewes et al., 1997), their use in Natural Language Processing is more recent. Fischer (2003) use string generating HRG to model discontinuous constituents in German. Jones et al. (2012) introduce SHRG and demonstrate an application to construct intermediate semantic representations in machine translation. Peng et al. (2015) automatically extract SHRG rules from corpora annotated with graph based meaning representations (Abstract Meaning Representation), using Markov Chain Monte Carlo techniques. They report competitive results on string-to-graph parsing. Braune et al. (2014) empirically compare SHRG to cascades of tree transducers as devices to translate English strings into reentrant semantic graphs. In agreement with the result we show more formally in this paper, they observe that, to generate graphs that contain a larger number of long-distance dependencies, a larger grammar with more nonterminals is needed, because the derivations of the grammar are limited to string CFG derivations.

Synchronous context free string-graph grammars have also been studied in the framework of Interpreted Regular Tree Grammar (Koller and Kuhlmann, 2011) using S-Graph algebras (Koller, 2015). In the TAG community, HRGs have been discussed by Pitsch (2000), who shows a construction to convert TAGs into HRGs. Finally, Joshi and Rambow (2003) discuss a version of TAG in which the *derived* trees are dependency trees, similar to the SHRG approach we present here.

To use string-generating HRG in practice we need a HRG parser. Chiang et al. (2013) present an efficient graph parsing algorithm. However, their implementation assumes that graph fragments are connected, which is not true for the grammar in section 3. On the other hand, since string-generation HRGs are similar to LCFRS, any LCFRS parser could be used. The relationship between the two parsing problems merits further investigation. Seifert and Fischer (2004) describe a parsing algorithm specificaly for string-generating HRGs.

Formal properties of dependency structures generated by lexicalized formalisms have been studied in detail by Kuhlmann (2010). He proposes measures for different types of non-projectivity in dependency structures, including *edge degree* (which is related to the degree of crossing dependencies we use in this paper), and *block degree*. A qualitative measure of dependency structures is well nestedness, which indicates whether there is an overlap between subtrees that do not stand in a dominance relation to each other. In future work, we would like to investigate how these measures relate to dependency structures generated by HRG derivations and SHRG derived graphs.

7 Conclusion

In this paper we investigated the capability of hyperedge replacement graph grammars (HRG) and synchronous string-to-graph grammar (SHRG) to generate dependency structures for non-projective phenomena. Using Dutch cross-serial dependencies as an example, we compared two different approaches: string-generating HRGs whose derivation trees can be interpreted as dependency structures, and string-to-graph SHRGs, whose can create dependency structures as their derived graphs.

We provided an example grammar for each case. The derivation tree of the HRG adequately reflected syntactic dependencies and the example grammar could in principle generate an arbitrary number of crossing dependencies. However, these derivation trees are unlabeled and cannot be extended to represent deeper semantic relationships (e.g semantic argument structure and coreference). For the string-to-graph SHRG, we saw that the derived graph of our grammar represented the correct dependencies for the example sentence, while the derivation tree did not.

The main observation of this paper is that, unlike the string-generating HRG, the string-to-graph SHRG was only able to generate a limited number of crossing dependencies. With each additional crossing edge in the example, we needed to add a new rule with a higher hyperedge type, increasing the order of the grammar. We argued that the reason for this is that the synchronous derivation for the input string and output graph is constrained to be a valid string CFG derivation. Analyzing this observation more formally, we showed a relationship between the order of the grammar and the maximum permitted number of edges crossing into the span of another edge.

An important conclusion is that, unless the correct syntactic dependencies are already local in the derivation, HRGs cannot derive dependency graphs with an arbitrary number of cross-serial dependencies. We take this to be a strong argument for using lexicalized formalisms in synchronous grammars for syntactic and semantic analysis, that can process at least a limited degree of non-projectivity, such as LTAG.

In future work, we are aiming to develop a lexicalized, synchronous string-to-graph formalisms of this kind. We would also like to relate our results to other measures of non-projectivity discussed in the literature. Finally, we hope to expand the results of this paper to other non-projective phenomena and to semantic graphs.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Linguistic Annotation Workshop*.

Fabiene Braune, Daniel Bauer, and Kevin Knight. 2014. Mapping between english strings and reentrant semantic graphs. In *Proceedings of LREC*, Reykjavik, Iceland.

David Chiang, Jacob Andreas, Daniel Bauer, Karl-Mortiz Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing graphs with hyperedge replacement grammars. In *Proceedings of ACL*, Sofia, Bulgaria.

Frank Drewes, Annegret Habel, and Hans-Jörg Kreowski. 1997. Hyperedge replacement graph grammars. In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, pages 95–162. World Scientific.

Joost Engelfriet and Linda Heyker. 1991. The string generating power of context-free hypergraph grammars. *Journal of Computer and System Sciences*, 43(2):328–360.

Ingrid Fischer. 2003. Modeling discontinuous constituents with hypergraph grammars. In *International Workshop on Applications of Graph Transformations with Industrial Relevance (AGTIVE)*, pages 163–169.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl-Moritz Hermann, and Kevin Knight. 2012. Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of COLING*, Mumbai, India. First authorship shared.

Aravind Joshi and Owen Rambow. 2003. A formalism for dependency grammar based on tree adjoining grammar. *Proceedings of the Conference on Meaning-Text Theory*, pages 207–216.

Alexander Koller and Marco Kuhlmann. 2011. A generalized view on parsing and translation. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 2–13. Association for Computational Linguistics.

Alexander Koller. 2015. Semantic construction with graph grammars. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS)*, pages 228–238.

Marco Kuhlmann. 2010. *Dependency Structures and Lexicalized Grammars: An Algebraic Approach*, volume 6270. Springer.

Manfred Nagl. 1979. A tutorial and bibliographical survey on graph grammars. In *Proceedings of the International Workshop on Graph-Grammars and Their Application to Computer Science and Biology*, pages 70–126, London, UK, UK. Springer-Verlag.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for amr parsing. In *Proceedings of CONLL*.

Gisela Pitsch. 2000. Hyperedge replacement and tree adjunction. In Anne Abeillè and Owen Rambow, editors, *Tree Adjoining Grammars*. CSLI.

Owen Rambow and Aravind Joshi. 1997. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In Leo Wanner, editor, *Recent Trends in Meaning-Text Theory*, pages 167–190. John Benjamins, Amsterdam and Philadelphia.

Sebastian Seifert and Ingrid Fischer. 2004. Parsing string generating hypergraph grammars. In *International Conference on Graph Transformations(ICGT)*, pages 352–367.

David J. Weir. 1992. Linear context-free rewriting systems and deterministic tree-walking transducers. In *Proceedings of ACL*, pages 136–143, Newark, Delaware, USA, June. Association for Computational Linguistics.