Synchronous Models of Language

Owen Rambow
CoGenTex, Inc.
840 Hanshaw Road, Suite 11
Ithaca, NY 14850-1589
owen@cogentex.com

Giorgio Satta
Dipartimento di Elettronica ed Informatica
Università di Padova
via Gradenigo, 6/A
I-35131 Padova, Italy
satta@dei.unipd.it

Abstract

In synchronous rewriting, the productions of two rewriting systems are paired and applied synchronously in the derivation of a pair of strings. We present a new synchronous rewriting system and argue that it can handle certain phenomena that are not covered by existing synchronous systems. We also prove some interesting formal/computational properties of our system.

1 Introduction

Much of theoretical linguistics can be formulated in a very natural manner as stating correspondences (translations) between layers of representation; for example, related interface layers LF and PF in GB and Minimalism (Chomsky, 1993), semantic and syntactic information in HPSG (Pollard and Sag, 1994), or the different structures such as c-structure and f-structure in LFG (Bresnan and Kaplan, 1982). Similarly, many problems in natural language processing, in particular parsing and generation, can be expressed as transductions, which are calculations of such correspondences. There is therefore a great need for formal models of corresponding levels of representation, and for corresponding algorithms for transduction.

Several different transduction systems have been used in the past by the computational and theoretical linguistics communities. These systems have been borrowed from translation theory, a subfield of formal language theory, or have been originally (and sometimes redundantly) developed. *Finite state transducers* (for an overview, see, e.g., (Aho and Ullman, 1972)) provide translations between regular languages. These devices have been popular in computational morphology and computational phonology since the early eighties (Koskenniemi, 1983; Kaplan and Kay, 1994), and more recently in parsing as well (see, e.g., (Gross, 1989; Pereira, 1991; Roche, 1993)). *Pushdown transducers* and *syntax directed translation schemata* (SDTS) (Aho and Ullman, 1969) translate between context-free languages and are therefore more powerful than finite state transducers. Pushdown transducers are a standard model for parsing, and have also been used (usually implicitly) in speech understanding. Recently, variants of SDTS have been proposed as models for simultaneously bracketing parallel corpora (Wu, 1995). Synchronization of tree adjoining grammars (TAGs) (Shieber and Schabes, 1990; Shieber, 1994) are even more powerful than the previous formalisms, and have been applied in machine translation (Abeillé, Schabes, and Joshi, 1990; Egedi and Palmer, 1994; Harbusch and Poller, 1994; Prigent, 1994), natural language generation (Shieber and Schabes, 1991), and theoretical syntax (Abeillé, 1994). The common underlying idea in all of these formalisms is to combine two generative devices through a pairing of their productions (or, in the case of the corresponding automata, of their transitions) in such a way that right-hand side nonterminal symbols in the paired productions are *linked*. The processes of derivation proceed synchronously in the two devices by applying the paired grammar rules only to linked nonterminals introduced previously in the derivation. The fact that the above systems all reflect the same translation technique has not always been recognized in the computational linguistics literature. Following (Shieber and Schabes, 1990) we will refer to the general approach as *synchronous rewriting*. While synchronous systems are becoming more and more popular, surprisingly little is known about the formal characteristics of these systems (with the exception of the finite-state devices).

In this paper, we argue that existing synchronous systems cannot handle, in a computationally attrac-

tive way, a standard problem in syntax/semantics translation, namely quantifier scoping. We propose a new system that provides a synchronization between two unordered vector grammars with dominance links (UVG-DL) (Rambow, 1994). The type of synchronization is closely based on a previously proposed model, which we will call "local" synchronization. We argue that this synchronous system can deal with quantifier scoping in the desired way. The proposed system has the *weak language preservation property*, that is, the defined synchronization mechanism does not alter the weak generative capacity of the formalism being synchronized. Furthermore, the *tree-to-forest translation* problem for our system can be solved in polynomial time; that is, given a derivation tree obtained according to one of the synchronized grammars, we can construct the forest of all the translated derivation trees in the other grammar, using a polynomial amount of time.

The structure of this paper is as follows. In Section 2, we introduce quantifier raising and review two types of synchronization and mention some new formal results. We introduce our new synchronous system in Section 3, and present our formal results and outline the proof techniques in Section 4.

2 Types of Synchronization

2.1 Quantifier Raising

We start by presenting an example which is based on transfer between a syntactic representation and a "semantic" representation of the scoping of quantified NPs. It is generally assumed that in English (and many other languages), quantified arguments of a verb can (in appropriate contexts) take scope in any possible order, and that this generalization extends to cases of embedded clauses (May, 1985).[1] For example, sentence (1) can have four possible interpretations (of the six possible orderings of the quantifiers, two pairs are logically equivalent), two of which are shown in (2).

(1) Every man thinks some official said some Norwegian arrived

(2) a. $\forall x$, x a man, $\exists y$, y an official, $\exists z$, z a Norwegian, x thinks y said z arrived

b. $\exists z$, z a Norwegian, $\exists y$, y an official, $\forall x$, x a man, x thinks y said z arrived

[1] We explicitly exclude from our analysis cases of quantified NPs embedded in NPs, and do not, of course, propose to develop a serious linguistic theory of quantifier scoping.

We give a simplified syntactic representation for (1) in Figure 1, and a simplified semantic representation for (2b) in Figure 2.

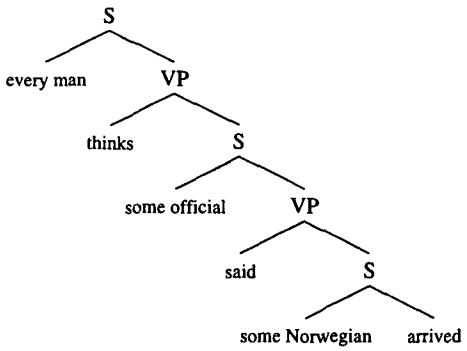

Figure 1: Syntactic representation for (1)

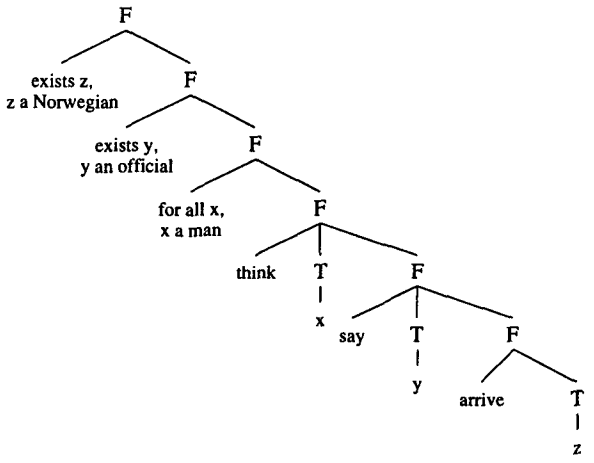

Figure 2: Semantic representation for (2b)

2.2 Non-Local Synchronization

We will first discuss a type of synchronization proposed by (Shieber and Schabes, 1990), based on TAG. We will refer to this system as *non-local synchronous TAG* (nlSynchTAG). The synchronization is non-local in the sense that once links are introduced during a derivation by a synchronized pair of grammar rules, they need not continue to impinge on the nodes that introduced them: the links may be reassigned to a newly introduced nonterminal when an original node is rewritten. We will refer to this mechanism as *link inheritance*. To illustrate, we will give as an example an analysis of the quantifier-raising example introduced above, extending in a natural manner an example given by Shieber and Schabes.

The elementary structures are shown in Figure 3 (we only give one NP — the others are similar). The nominal arguments in the syntax are associated with

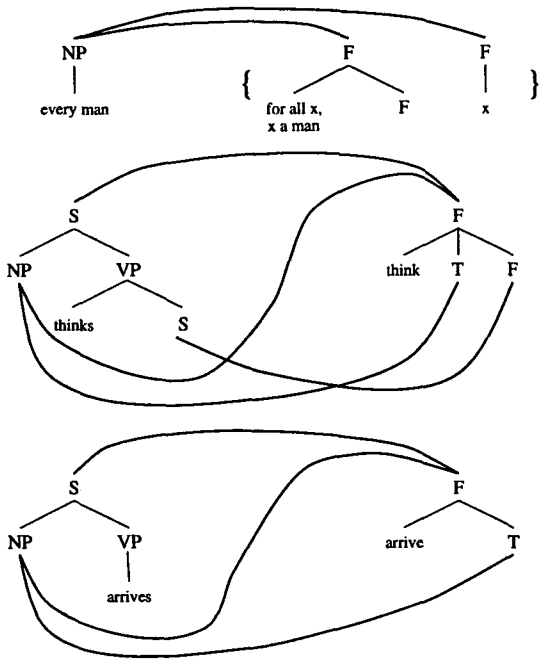

Figure 3: Elementary structures in nlSynchTAG

pairs of trees in the semantics, and are linked to two nodes, the quantifier and the variable. The derivation proceeds as illustrated in Figure 4, finally yielding the two structures in Figure 1 and Figure 2. Note that some of the links originating with the NP nodes are inherited during the derivation. By changing the order in which we add the nominal arguments at the end of the derivation, we can obtain all quantifier scopes in the semantics.

The problem with non-local synchronization is that the weak language preservation property does not hold. (Shieber, 1994) shows that not all nlSynchTAG left-projection languages can be generated by TAGs. As a new result, in (Rambow and Satta, 1996) we show that the recognition of some fixed left-projection languages of a nlSynchTAG is NP-complete. Our reduction crucially relies on link inheritance. This makes nlSynchTAG unattractive for applications in theoretical or computational linguistics.

2.3 Local Synchronous Systems

In contrast with non-local synchronization, in local synchronization there is no inheritance of synchronization links. This is enforced by requiring that the links establish a bijection between nonterminals in the two synchronously derived sentential forms, that is, each nonterminal must be involved in exactly one link. In this way, once a nonterminal is rewritten through the application of a pair of rules to two

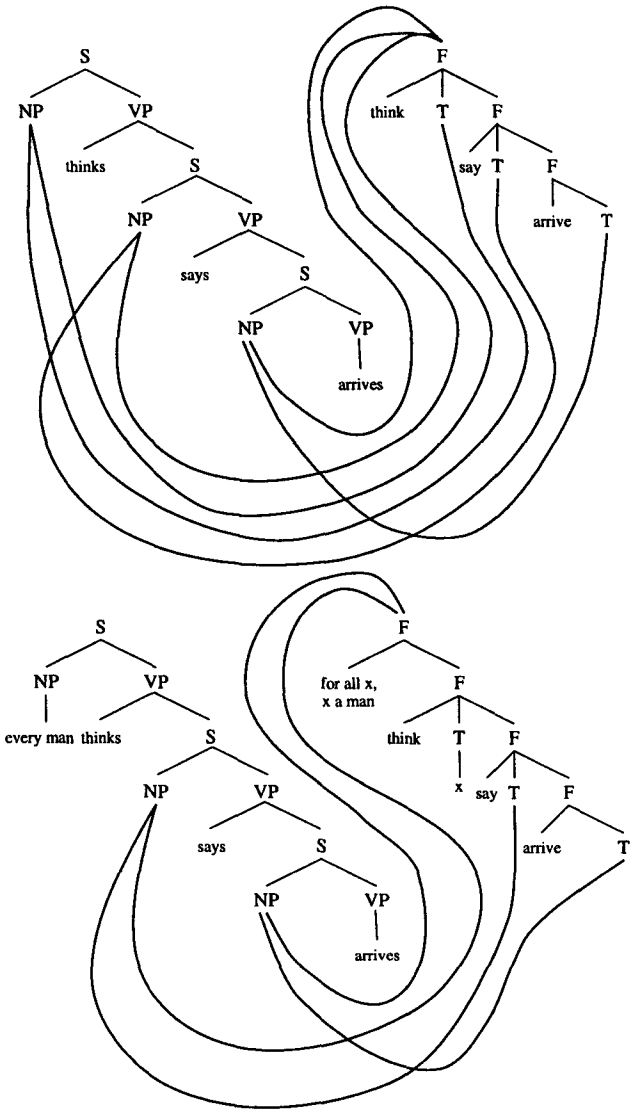

Figure 4: Non-local derivation in nlSynchTAG

linked nonterminals, no additional link remains to be transferred to the newly introduced nonterminals. As a consequence of this, the derivation structures in the left and right grammars are always isomorphic (up to ordering and labeling of nodes).

The canonical example of local synchronization is SDTS (Aho and Ullman, 1969), in which two context-free grammars are synchronized. We give an example of an SDTS and a derivation in Figure 5. The links are indicated as boxed numbers to the right of the nonterminal to which they apply. (Shieber, 1994) defines the tree-rewriting version of SDTS, which we will call *synchronous TAG* (SynchTAG), and argues that SynchTAG does not have the formal problems of nlSynchTAG (though

Grammar:

$S_1 \rightarrow NP_2$ *likes* NP_3 | $S_1 \rightarrow NP_3$ *plaît à* NP_2

$NP_4 \rightarrow$ *John* | $NP_4 \rightarrow$ *Jean*

$NP_5 \rightarrow$ *the white* N_6 | $NP_5 \rightarrow$ *la* N_6 *blanche*

$N_7 \rightarrow$ *house* | $N_7 \rightarrow$ *maison*

Derivation:

(S_0, S_0)

$\Rightarrow (NP_2$ *likes* NP_3, NP_3 *plaît à* $NP_2)$

$\Rightarrow (NP_2$ *likes the white* N_6, *la* N_6 *blanche plaît à* $NP_2)$

$\overset{*}{\Rightarrow}$ (*John likes the white house, la maison blanche plaît à Jean*)

Figure 5: Sample SDTS and derivation

(Shieber, 1994) studies the translation problem making the unappealing assumption that each tree in the input grammar is associated with only one output grammar tree).

However, SynchTAG cannot derive all possible scope orderings, because of the locality restriction. This can be shown by adapting the proof technique in (Becker, Rambow, and Niv, 1992). In the following section, we will present a synchronous system which has local synchronization's formal advantages, but handles the scoping data.

3 Extended Local Synchronization

In this section, we propose a new synchronous system, which is based on local synchronization of unordered vector grammars with dominance links (UVG-DL) (Rambow, 1994). The presentations will be informal for reasons of space; we refer to (Rambow and Satta, 1996) for details. In UVG-DL, several context-free string rewriting rules are grouped into sets, called *vectors*. In a derivation, all or no rules from a given instance of a vector must be used. Put differently, all productions from a given vector must be used the same number of times. They can be applied in any order and need not be applied simultaneously or one right after the other. In addition, UVG-DL has *dominance links*. An occurrence of a nonterminal A in the right-hand side of a rule p can be linked to the left-hand nonterminal of another rule p' in the same vector. This dominance link will act as a constraint on derivations: if p is used in a derivation, then p' must be used subsequently in the subderivation that starts with the occurrence of A introduced by p. A UVG-DL is *lexicalized* iff at least one production in every vector contains a terminal symbol. Henceforth, all UVG-DLs mentioned in this paper will implicitly be assumed to be lexicalized. The derivation structure of a UVG-DL is just the derivation structure of the same derivation in the underlying context-free grammar (the CFG obtained by forming the union of all vectors). We give an example of a UVG-DL in Figure 6, in which the dotted lines represent the dominance links. A sample derivation is in Figure 7.

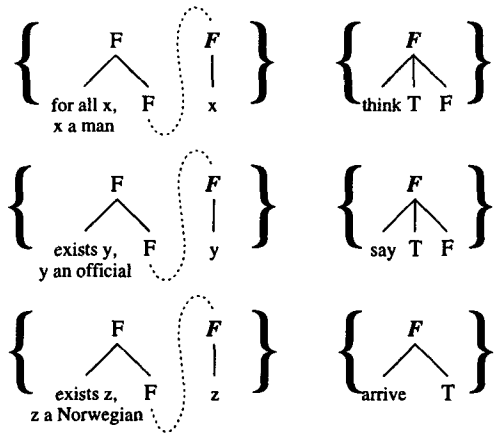

Figure 6: A UVG-DL for deriving semantic representations such as (2)

Our proposal for the synchronization of two UVG-DL uses the notion of locality in synchronization, but with respect to entire vectors, not individual productions in these vectors. This approach, as we will see, gives us both the desired empirical coverage and acceptable computational and formal results. We suppose that in each vector v of a UVG-DL there is exactly one privileged element, which we call the *synchronous production* of v. All other elements of v are referred to as *asynchronous productions*. In Figures 6 and 7, the synchronous productions are designated by a bold-italic left-hand side symbol. Furthermore, in the right-hand side of each asynchronous production of v we identify a single nonterminal nonterminal, called the *heir*.

In a *synchronous UVG-DL* (SynchUVG-DL), vectors from one UVG-DL are synchronized with vectors from another UVG-DL. Two vectors are synchronized by specifying a bijective synchronization mapping (as in local synchronization) between the non-heir right-hand side occurrences of nonterminals in the productions of the two vectors. A nonterminal on which a synchronization link impinges is referred to as a *synchronous nonterminal*. A sample SynchUVG-DL grammar is shown in Figure 9.

Informally speaking, during a SynchUVG-DL derivation, the two synchronous productions in a pair of synchronized vectors must be applied at the same time and must rewrite linked occurrences of nonterminals previously introduced. The asynchronous productions of the two synchronized gram-

mars are not subject to the synchronization requirement, and they can be applied at any time and independently of the other grammar (but of course subject to the grammar-specific dominance links). Any synchronous links that impinge on a nonterminal rewritten by an asynchronous production are transferred to the heir of the asynchronous production. A production may introduce a synchronous nonterminal whose counterpart in the other grammar has not yet been introduced. In this case, the link remains "pending". Thus, while in SynchUVG-DL there is link inheritance as in non-local synchronization, link inheritance is only possible with those productions that themselves are not subject to the synchronization requirement.

The locality of the synchronization becomes clear when we consider a new tree structure which we introduce here, called the *vector derivation tree.* Consider two synchronized UVG-DLderivations in a SynchUVG-DL. The vector derivation tree for either component derivation is obtained as follows. Each instance of a vector used in the derivation is represented as a single node (which we label with that vector's lexeme). A node representing a vector v_1 is immediately dominated by the node representing the vector v_2 which introduced the synchronization link that the synchronous production of v_1 rewrites. Unlike the standard derivation tree for UVG-DL, the vector derivation tree clearly shows how the vectors (rather than the component rules of the vectors) were combined during the derivation. The vector derivation tree for the derivation in Figure 7 is shown in Figure 8.

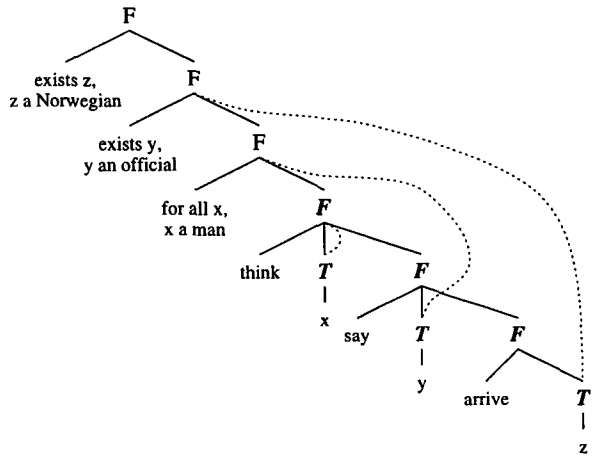

Figure 7: Derivation of (2b) in a UVG-DL

It should be clear that the vector derivation trees for two synchronized derivations are isomorphic, reflecting the fact that our definition of SynchUVG-

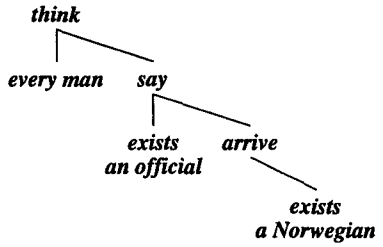

Figure 8: Vector derivation tree for derivation of (2b)

DL is local with respect to vectors (though not with respect to productions, since the derivation trees of two synchronized UVG-DL derivations need not be isomorphic). The vector derivation tree can be seen as representing an "outline" for the derivation. Such a view is attractive from a linguistic perspective: if each vector represents a lexeme and its projection (where the synchronous production is the basis of the lexical projection that the vector represents), then the vector derivation tree is in fact the dependency tree of the sentence (representing direct relations between lexemes such as grammatical function). In this respect, the vector derivation tree of UVG-DL is like the derivation tree of tree adjoining grammar and of D-tree grammars (DTG) (Rambow, Vijay-Shanker, and Weir, 1995), which is not surprising, since all three formalisms share the same extended domain of locality. Furthermore, the vector derivation tree of SynchUVG-DL shares with the the derivation tree of DTG the property that it reflects linguistic dependency uniformly; however, while the definition of DTG was motivated precisely from considerations of dependency, the vector derivation tree is merely a by-product of our definition of SynchUVG-DL, which was motivated from the desire to have a computationally tractable model of synchronization more powerful than SynchTAG.[2]

We briefly discuss a sample derivation. We start with the two start symbols, which are linked. We then apply an asynchronous production from the semantic grammar. In Figure 10 (top) we see how the link is inherited by the heir nonterminal of the applied production. This step is repeated with two more asynchronous productions, yielding Figure 10 (bottom). We now apply productions for the bodies of the clauses, but stop short before the two synchronous productions for the *arrive* clause, yielding Figure 11. We see the asynchronous production of the syntactic *arrive* vector has not only inherited the link to its heir nonterminal, but has introduced a link

[2]We do not discuss modifiers in this paper for lack of space.

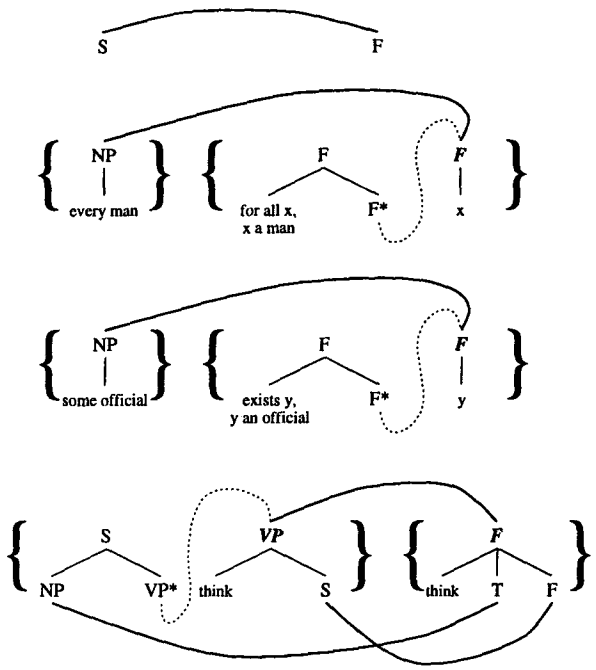

Figure 9: SynchUVG-DL grammar for quantifier scope disambiguation

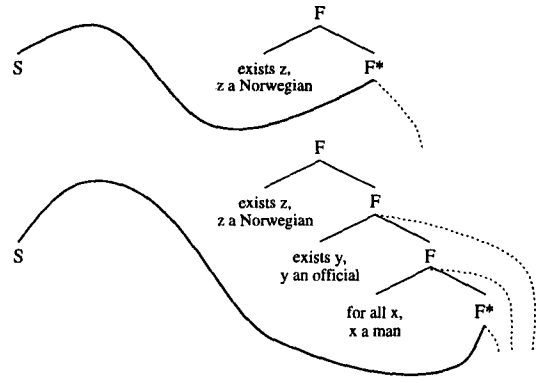

Figure 10: SynchUVG-DL derivation, steps 1 and 2

of its own. Since the semantic end of the link has not been introduced yet, the links remains "pending" until that time. We then finish the derivation to obtain the two trees in Figure 1 and Figure 2, with no synchronization or dominance links left.

4 Formal results

Theorem 1 SynchUVG-DL *has the language preservation property.*

Proof (outline). Let G_s be a SynchUVG-DL, G' and G'' its left and right UVG-DL components, respectively. We construct a UVG-DL G generating the left-projection language of G_s. G uses all the

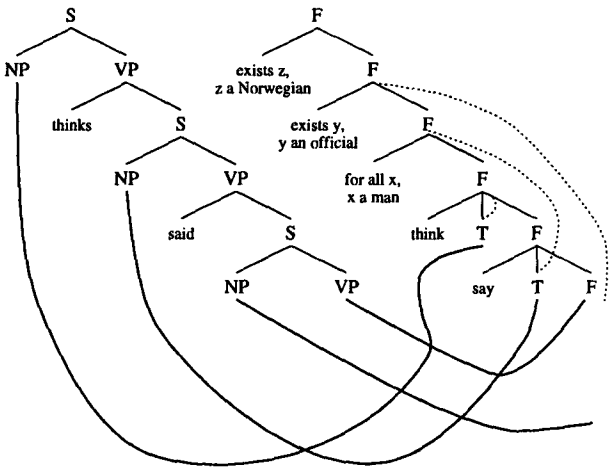

Figure 11: SynchUVG-DL derivation, step 3

nonterminal symbols of G' and G'', and some compound nonterminals of the form $[A, B]$, A and B nonterminals of G' and G'', respectively. G simulates G_s derivations by intermixing symbols of G' and symbols of G'', and without generating any of the terminal symbols of G''. Most important, each pair of linked nonterminals generated by G_s is represented by G using a compound symbol. This enforces the requirement of simultaneous application of synchronous productions to linked nonterminals.

Each vector v of G is constructed from a pair of synchronous vectors (v', v'') of G_s as follows. First, all instances of nonterminals in v'' are replaced by ε. Furthermore, for any instance B of a right-hand side nonterminal of v'' linked to a right-hand side nonterminal A of v', B is replaced by ε and A by $[A, B]$. Then the two synchronous productions in v' and v'' are composed into a single production in v, by composing the two left-hand sides in a compound symbol and by concatenating the two right-hand sides. Finally, to simulate link inheritance in derivations of G_s, each asynchronous production in v' and v'' is transferred to v, either without any change, or by composing with some nonterminal C both its left-hand side and the heir nonterminal in its right-hand side. Note that there are finitely many choices for the last step, and each choice gives a different vector in G, simulating the application of v' and v'' to a set of (occurrences of) nonterminals in a particular link configuration in a sentential form of G_s. ∎

We now introduce a representation for sets of derivation trees in a UVG-DL G. A *parse tree* in G is an ordered tree representing a derivation in G and encoding at each node the production p used to start the corresponding subderivation and the multiset of productions f used in that subderivation. A

parse forest in G is a directed acyclic graph which is ordered and bipartite. (We use ideas originally developed in (Lang, 1991) for the context-free case.) Nodes of the graph are of two different types, called *and-nodes* and *or-nodes*, respectively, and each directed arc connects nodes of different types. A parse forest in G represents a set T of parse trees in G if the following holds. When starting at a root node and walking through the graph, if we follow exactly one of the outgoing arcs at each or-node, and all of the outgoing arcs at each and-node, we obtain a tree in T modulo the removal of the or-nodes. Furthermore, every tree in T can be obtained in this way.

Lemma 2 *Let G be a* UVG-DL *and let $q \geq 1$ be a natural number. The parse forest representing the set of all parse trees in G with no more than q vectors can be constructed in an amount of time bounded by a polynomial function of q.* ■

Let G_s be a SynchUVG-DL, G' and G'' its left and right UVG-DL components, respectively. For a parse tree τ in G', we denote as $T(\tau)$ the set of all parse trees in G'' that are synchronous with τ according to G_s. The *parse-to-forest translation* problem for G_s takes as input a parse tree τ in G' and gives as output a parse forest representation for $T(\tau)$. If G_s is lexicalized, such a parse forest has size bounded by a polynomial function of $|\tau|$, despite the fact that the size of $T(\tau)$ can be exponentially larger than the size of τ. In fact, we have a stronger result.

Theorem 3 *The parse-to-forest translation problem for a lexicalized* SynchUVG-DL *can be computed in polynomial time.*

Proof (outline). Let G_s be a SynchUVG-DL with G' and G'' its left and right UVG-DL components, respectively. Let τ be a parse tree in G' and π be the parse forest representing $T(\tau)$. The construction of π consists of two stages.

In the first stage, we construct the vector derivation tree γ associated with τ. Let q be the number of nodes of γ. We also construct a parse forest π_q representing the set of all parse trees in G'' with no more than q vectors. This stage takes polynomial time in the size of τ, since γ can be constructed from τ in linear time and π_q can be constructed as in Lemma 2.

In the second stage, we remove from π_q all the parse trees not in π. This completes the construction, since the set of parse trees represented by π is included in the set of parse trees represented by π_q. Let n_r and Γ be the root node and the set of all nodes of γ, respectively. For $n \in \Gamma$, $out(n)$ denotes the set of all children of n. We call *family* the set $\{n_r\}$ and any nonempty subset of $out(n)$, $n \in \Gamma$. The main idea is to associate a set of families $\mathcal{F}_n$ to each node n of π_q, such that the following condition is satisfied. A family F belongs to $\mathcal{F}_n$ if and only if at least one subderivation in G'' represented at n induces a forest of vector derivation trees whose root nodes are all and only the nodes in F. Each $\mathcal{F}_n$ can easily be computed visiting π_q in a bottom-up fashion. Crucially, we "block" a node of π_q if we fail in the construction of $\mathcal{F}_n$. We claim that each set $\mathcal{F}_n$ has size bounded by the number of nodes in γ. This can be shown using the fact that all derivation trees represented at a node of π_q employ the same multiset of productions of G''. From the above claim, it follows that π_q can be processed in time polynomial in the size of τ. Finally, we obtain π simply by removing from π_q all nodes that have been blocked. ■

5 Conclusion

We have presented SynchUVG-DL, a synchronous system which has restricted formal power, is computationally tractable, and which handles the quantifier-raising data. In addition, SynchUVG-DL can be used for modeling the syntax of languages with syntactic constructions which have been argued to be beyond the formal power of TAG, such as scrambling in German and many other languages (Rambow, 1994) or *wh*-movement in Kashmiri (Rambow, Vijay-Shanker, and Weir, 1995). SynchUVG-DL can be used to synchronize a syntactic grammar for these languages either with a semantic grammar, or with the syntactic grammar of another language for machine translation applications. However, SynchUVG-DL cannot handle the list of cases listed in (Shieber, 1994). These pose a problem for SynchUVG-DL for the same reason that they pose a problem for other local synchronous systems: the (syntactic) dependency structures represented by the two derivations are different. These cases remain an open research issue.

Acknowledgments

Parts of the present research were done while Rambow was supported by the North Atlantic Treaty Organization under a Grant awarded in 1993, while at TALANA, Université Paris 7, and while Satta was visiting the Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD.

References

Abeillé, Anne. 1994. Syntax or semantics? Handling nonlocal dependencies with MCTAGs or

Synchronous TAGs. *Computational Intelligence*, 10(4):471–485.

Abeillé, Anne, Yves Schabes, and Aravind Joshi. 1990. Using lexicalized TAGs for machine translation. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING'90)*, Helsinki. COLING-90.

Aho, A. V. and J. D. Ullman. 1969. Syntax directed translations and the pushdown assembler. *J. Comput. Syst. Sci.*, 3(1):37–56.

Aho, A. V. and J. D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling.* Prentice Hall, Englewood Cliffs, NJ.

Becker, Tilman, Owen Rambow, and Michael Niv. 1992. The derivational generative power, or, scrambling is beyond LCFRS. Technical Report IRCS-92-38, Institute for Research in Cognitive Science, University of Pennsylvania.

Bresnan, J. and R. Kaplan. 1982. Lexical-functional grammar: A formal system for grammatical representation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations.* MIT Press.

Chomsky, Noam. 1993. A minimalist program for linguistic theory. In Kenneth Hale and Samuel J. Keyser, editors, *The View from Building 20.* MIT Press, Cambridge, Mass., pages 1–52.

Egedi, Dana and Martha Palmer. 1994. Constraining lexical selection across languages using TAG. In *3e Colloque International sur les Grammaires d'Arbres Adjoints (TAG+3)*, Rapport Technique TALANA-RT-94-01. Université Paris 7.

Gross, Maurice. 1989. The use of Finite-State Automata in the lexical representation of natural language. In M. Gross and D. Perrin, editors, *Electronic Dictionaries and Automata in Computational Linguistics.* Springer.

Harbusch, Karin and Peter Poller. 1994. Structural rewriting with synchronous rewriting systems. In *3e Colloque International sur les Grammaires d'Arbres Adjoints (TAG+3)*, Rapport Technique TALANA-RT-94-01. Université Paris 7.

Kaplan, Ronald M. and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378.

Koskenniemi, Kimmo. 1983. Two-level morphology: A general computational model for word-form recognition and production. Technical Report 11, Department of General Linguistics, University of Helsinki.

Lang, B. 1991. Towards a uniform formal framework for parsing. In M. Tomita, editor, *Current Issues in Parsing technology.* Kluwer Academic Publishers, chapter 11, pages 153–171.

May, Robert. 1985. *Logical Form: Its structure and Derivation.* MIT Press, Cambridge, Mass.

Pereira, Fernando. 1991. Finite-state approximation of phrase structure grammars. In *29th Meeting of the Association for Computational Linguistics (ACL'91)*, Berkeley, California. ACL.

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

Prigent, Gilles. 1994. Synchronous tags and machine translation. In *3e Colloque International sur les Grammaires d'Arbres Adjoints (TAG+3)*, Rapport Technique TALANA-RT-94-01. Université Paris 7.

Rambow, Owen. 1994. Multiset-valued linear index grammars. In *32nd Meeting of the Association for Computational Linguistics (ACL'94).* ACL.

Rambow, Owen and Giorgio Satta. 1996. Synchronous models of language. Manuscript under preparation.

Rambow, Owen, K. Vijay-Shanker, and David Weir. 1995. D-Tree Grammars. In *33rd Meeting of the Association for Computational Linguistics (ACL'95).* ACL.

Roche, Emmanuel. 1993. *Analyse syntaxique transformationelle du français par transducteur et lexique-grammaire.* Ph.D. thesis, Université Raris 7, Paris, France.

Shieber, Stuart and Yves Schabes. 1990. Synchronous tree adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*, Helsinki.

Shieber, Stuart and Yves Schabes. 1991. Generation and synchronous tree adjoining grammars. *Computational Intelligence*, 4(7):220–228.

Shieber, Stuart B. 1994. Restricting the weak generative capacity of Synchronous Tree Adjoining Grammar. *Computational Intelligence*, 10(4):371–385.

Wu, Dekai. 1995. An algorithm for simultaneously bracketing parallel texts by aligning words. In *33rd Meeting of the Association for Computational Linguistics (ACL'95).* ACL.