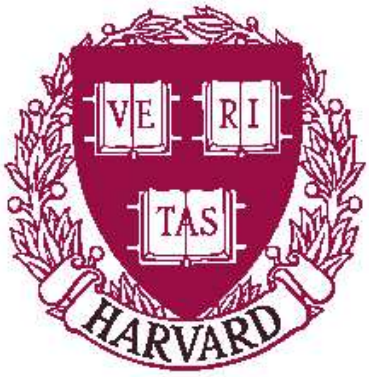
Induction of Probabilistic Synchronous Tree-Insertion Grammars

Rebecca Nesson
Stuart M. Shieber
and
Alexander Rush

TR-20-05



Computer Science Group
Harvard University
Cambridge, Massachusetts

INDUCTION OF PROBABILISTIC SYNCHRONOUS TREE-INSERTION GRAMMARS

REBECCA NESSON, STUART M. SHIEBER, AND ALEXANDER RUSH

Abstract. *Draft. Comments weolcomed. Please do not cite or quote without prior consent. Revision 1.9 of August 15, 2005, 15:45:21, generated November 4, 2005.*

Increasingly, researchers developing statistical machine translation systems have moved to incorporate syntactic structure in the models that they induce. These researchers are motivated by the intuition that the limitations in the finite-state translation models exemplified by IBM's "Model 5" follow from the inability to use phrasal and hierarchical information in the interlingual mapping. What is desired is a formalism that has the *substitution*-based hierarchical structure provided by context-free grammars, with the *lexical* relationship potential of n-gram models, with processing *efficiency* no worse than CFGs. Further, it should ideally allow for *discontinuity* in phrases, and be *synchronizable*, to allow for multilinguality. Finally, in order to support automated induction, it should allow for a *probabilistic* variant. We introduce probabilistic synchronous tree-insertion grammars (PSTIG) as such a formalism. In this paper, we define a restricted version of PSTIG, and provide algorithms for parsing, parameter estimation, and translation. As a proof of concept, we successfully apply these algorithms to a toy problem, corpus-based induction of a statistical translator of arithmetic expressions from postfix to partially parenthesized infix.

1. Introduction

Increasingly, researchers developing statistical machine translation systems have moved to incorporate syntactic structure in the models that they induce. These researchers are motivated by the intuition that the limitations in the finite-state translation models exemplified by IBM's "Model 5" (Brown, Pietra, Pietra, and Mercer, 1993) follow from the inability to use phrasal and hierarchical information in the interlingual mapping. It is suggestive that bilingual dictionaries describe the mappings between languages in terms of constructions, not individual words. For instance, the *HarperCollins Italian College Dictionary* (HCICD) translates the English "to take advantage of" as "sfruttare", although that word is a direct translation of neither "take" nor "advantage".

IBM-style models can be augmented to allow multiword (in addition to single word) mappings. Marcu and Wong (2002) use joint probability distributions over frequently co-occurring n-grams to find multiword translations, thereby improving on the performance of IBM Model 5. Such an approach does allow multiword relationships to be induced, but does not in any sense incorporate syntactic structure to do so. Indeed, the natural way to augment the multiword approach to incorporate syntactic constraints is to restrict the multiword sequences to syntactic constituents (as determined by a statistical parser for instance) (Yamada and

Knight, 2002). This augmentation turns out to underperform the syntax-free variant (Koehn, Och, and Marcu, 2003).

The reason is not hard to understand: The word sequences that map well in translation — such as the German-English example of Koehn et al. (2003) "es gibt"/"there is" — are not themselves syntactic constituents, but rather syntactic templates ("es gibt ..."/"there is ...") with "holes" (marked here by ellipses) that might be substituted for in some uniform manner. Bilingual dictionaries even make the mapping between such holes explicit through the use of place fillers like "sb" ("somebody"), "sth" ("something"), "qn" ("qualcuno"), etc., as in the HCICD entry "to drive sb mad"/"far impazzire qn". Secondarily, the phrases that are mapped need not appear contiguously, either because the holes split the lexical material, as in the example "drive sb mad", or because other constituents interpose themselves, as in the phrase "take advantage yesterday of sb".[1]

This ability to substitute subparts is, of course, the hallmark of context-free grammars. A natural approach, then, is to incorporate some sort of synchronization of context-free structures to allow for these kinds of mappings. This idea has a long history, starting with syntax-directed translation schemata (P. M. Lewis and Stearns, 1968; Aho and Ullman, 1969), and most methodically developed for natural-language processing purposes in Melamed's work on multitext grammars (Melamed, 2003, 2004). It appears in several variants, including inversion transduction grammars (Wu, 1996, 1997) and head transducers (Alshawi, Bangalore, and Douglas, 2000). Though the incorporation of substitution is well-motivated, the systems manifest other problems following from the type of structures that they synchronize. In particular, probabilistic context-free grammars (PCFG) are well known to perform poorly as language models compared to the syntactically stunted finite-state models; they gain the ability to substitute according to abstract categories at the expense of stating lexical relationships directly. For the same reason, synchronizing context-free grammars loses the lexical dependencies so crucial for translation, and so well characterized by the finite-state approach.

What is desired, then, is a formalism that has the *substitution*-based hierarchical structure provided by context-free grammars, with the *lexical* relationship potential of n-gram models, with processing *efficiency* no worse than CFGs. Further, it should ideally allow for *discontinuity* in phrases, and be *synchronizable*, to allow for multilinguality. Finally, in order to support automated induction, it should allow for a *probabilistic* variant.

Fortunately, such a formalism exists: probabilistic synchronous tree-insertion grammars (PSTIG). In this paper, we define a version of PSTIG, and provide algorithms for parsing, parameter estimation, and translation. As a proof of concept, we successfully apply these algorithms to a toy problem, corpus-based induction of a statistical translator of arithmetic expressions from postfix to partially parenthesized infix.

We begin by reviewing synchronous grammars in general and deriving synchronous tree-insertion grammar (STIG) in Section 2 motivated by the desiderata above. We then describe a deductive parsing algorithm for a restricted version of STIG Section 3. In Section 4, we present a probabilistic version of STIG, along with

[1]For example, "The US took advantage yesterday of the political and military momentum in its Afghan campaign. ..." is one of many Google hits on the phrase "took advantage yesterday of".

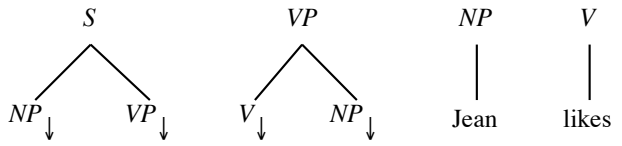


FIGURE 1. Context-free grammar in the form of unit height trees.

versions of the inside-outside algorithm and the EM algorithm for parameter estimation. Finally, we present preliminary empirical results on the toy problem in Section 5.

2. Synchronous Tree-Insertion Grammars

Synchronous grammar formalisms generalize a base formalism by *pairing* the elementary structures (the productions, rules, or elementary trees) of the base formalism, and *linking* nodes between paired structures. The operations on elementary structures are then generalized to operations on pairs of elementary structures restricted to act at linked nodes.

A simple example of the notion is synchronous context-free grammars. We view the elementary structures of a context-free grammar not as a production (e.g., $S \rightarrow NP\ VP$) but rather a tree of unit height, for example, those shown in Figure 1. The primitive operation is substitution of a tree rooted in a nonterminal A for a frontier node labeled with the same nonterminal A. (Frontier nodes subject to substitution, *substitution nodes*, are marked with a diacritic "↓". This annotation ceases to be redundant in later developments.) We generalize a context-free elementary tree t to a synchronous elementary tree pair $\langle t_L, t_R, \frown \rangle$ where the two trees t_L and t_R are context-free grammar rules and $\frown$ is a set of links specifying pairs of linked substitution nodes from t_L and t_R. Derivations proceed just as in a context-free grammar except that all nodes linked by some link in $\frown$ are simultaneously substituted for by paired trees derived by the grammar. The grammar fragment shown in Figure 2 gives an example of a synchronous context-free grammar that could be used to parse the English and French sentences "Jean likes red candies"/"Jean aime les bonbons rouges". (In this figure and elsewhere, we mark the linking relation graphically directly on the nodes rather than as a separate element in the triple.) The formalism allows the grammar to express correspondences between constituents in the two languages at a level higher than just single words, while also expressing differences in the ordering of constituents. This follows the intuition that constituent membership provides more useful information than a simple word distance metric when determining which words align in a multilingual sentence pair.[2]

[2]The parsing of string pairs according to an SCFG where the elementary trees are restricted to Chomsky normal form can be done in $O(n^6)$ time where n is the string length. We assert this without proof here. Informally, however, the parsing algorithm works by keeping indices of the ends of the substring covers in each string — two for each string — as well as an index for each string between the two endpoints on which to try to split the substring into its constituent parts, that is, six string indices. Iteration over all possible combinations of the six indices is thus $O(n^6)$. However, the restriction to Chomsky normal form limits the ability of SCFGs to express many different crossing correspondences and discontinuous constituents. Allowing more nonterminal

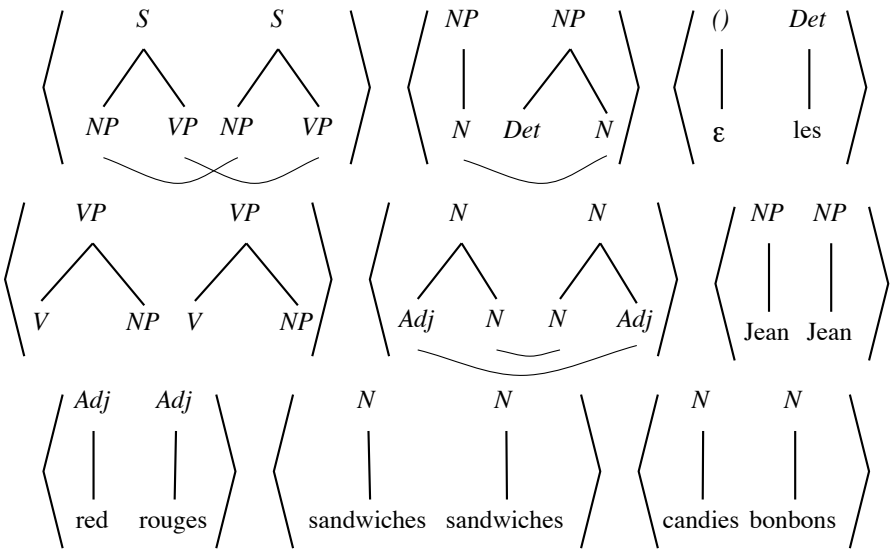


FIGURE 2. A partial synchronous CFG grammar for English and French represented in tree form

Although SCFGs solve the substitution problem, they introduce other difficulties. Primarily, they give up one of the main advantages of the IBM-style systems: the direct expression of relationships between individual words. Consider again the SCFG shown in Figure 2. The derivation for the sentence pair "Jean likes red sandwiches"/"Jean aime les sandwiches rouges" differs from that of the sentence pair "Jean likes red candies"/"Jean aime les bonbons rouges" only in the substitution of the particular elementary tree pair at the N-link in the fifth elementary tree pair. Thus, any distinction in the quality of the two sentence pairs must depend only on the nouns and not, for instance, on the lexical environment, in particular, on the adjacent word "red"/"rouges". This shortcoming could be remedied by *lexicalizing* the grammar, that is, by requiring that each elementary tree have a lexical item present, thereby making it possible for operations to express direct relationships between words. In general, however, context-free grammars cannot be lexicalized while still preserving the trees produced (Schabes and Waters, 1995). In addition, CFGs do not cleanly model the optional modification required for phrases such as "take advantage yesterday of" because they require additional nonterminal symbols in each place at which an optional modification might occur.

This problem is not inherently a problem of *synchronous* context-free grammars, but of context-free grammars in general: they cannot be lexicalized directly. Tree-adjoining grammars (TAG), introduced in monolingual form by Joshi (1985), and in a synchronous variant (STAG) by Shieber and Schabes (1990), are natural choices to capture lexically-based dependencies without losing the expressivity of

---

symbols and unrestricted links on the right-hand side of rules can increase expressivity at the expense of computation time. This tradeoff exists in all synchronous formalisms discussed in this paper.

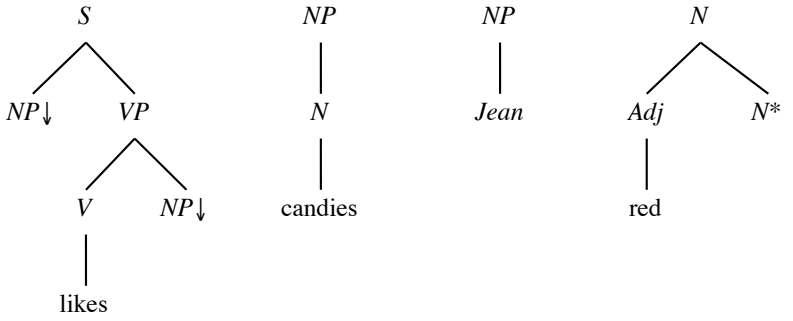


FIGURE 3. Sample TAG Elementary Trees

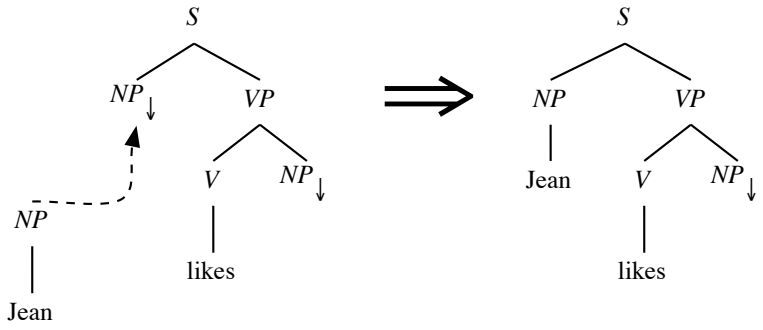


FIGURE 4. An Example TAG substitution

CFGs. Tree-adjoining grammars extend context-free grammars in two ways. First, the elementary trees are not restricted to unit height; the elementary trees can be of arbitrary size. Second, an additional operation beyond substitution is allowed, *adjunction*. Auxiliary trees are elementary trees in which the root and a distinguished frontier node, the *foot node*, are labeled with the same nonterminal. (By convention, the foot node is marked with a diacritic "$*$". The path from root to foot is called the *spine*.) The adjunction operation involves splicing an *auxiliary* tree with root and designated foot node labeled with a nonterminal A at a node in an elementary tree also labeled with nonterminal A. Figure 3 shows some examples of elementary trees. Examples of the substitution and adjunction operations are shown in Figures 4 and 5.

Importantly, Schabes, Abeille, and Joshi (1988) show that tree-adjoining grammars can lexicalize context-free grammars without changing the trees produced. Because each elementary tree contains a lexical item, the operations of substitution and adjunction implicitly manifest a lexical relationship. In addition, the two operations of TAG, substitution and adjunction, are exactly what is needed to handle the noncontiguity problems exemplified above, as shown in Figures 6 and 7.

However, the TAG formalism's additional expressivity leads to additional processing complexity. TAG parsing requires $O(n^6)$ time; synchronous TAG parsing

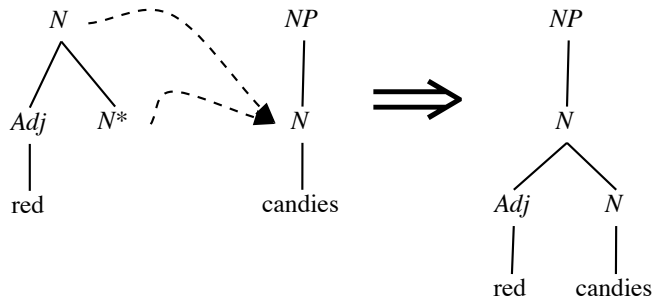


FIGURE 5. An Example TAG adjunction

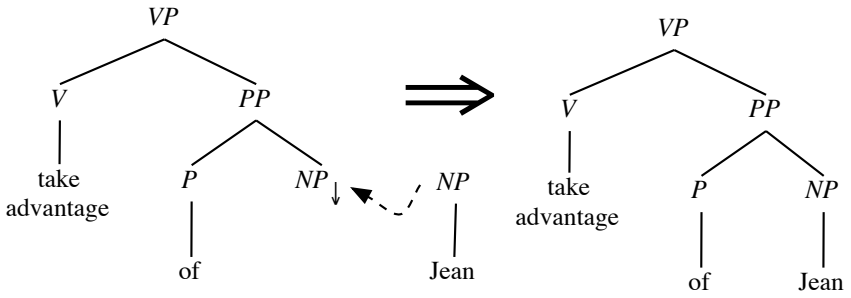


FIGURE 6. A TIG substitution to form the construction "take advantage of Jean"

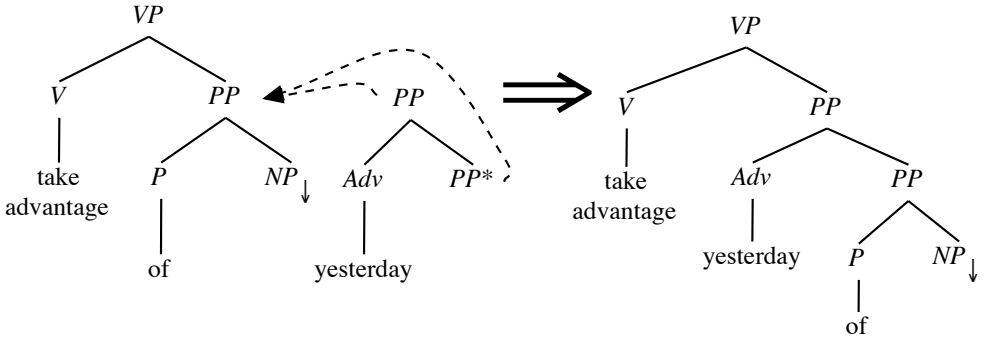


FIGURE 7. A TIG adjunction to form the construction "take advantage yesterday of"

would therefore require $O(n^{12})$ time. Because training of an MT system based on synchronous TAG would require repeated parsing of the training corpus, the time complexity of parsing synchronous TAG is prohibitive.

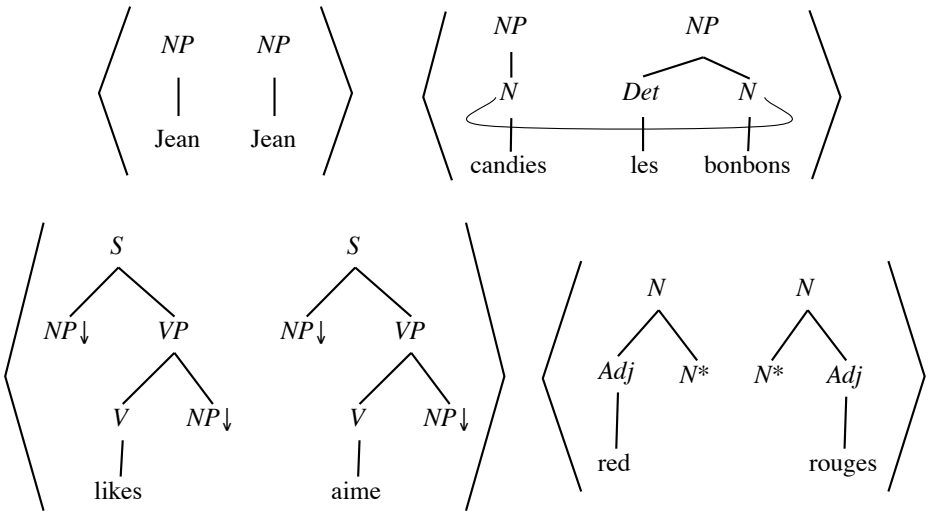


FIGURE 8. An STIG for an English/French version of our sample grammar

Tree-insertion grammars (TIG) are a computationally attractive alternative to TAG (Schabes and Waters, 1993). TIGs are similar to TAGs except that restrictions are placed on the form of elementary trees and on the adjunction operation. In particular, the foot node of an auxiliary tree is required to be at the left or right edge of the frontier, so that all textual material dominated by the spine will fall to the right or left, respectively, of the foot. The auxiliary trees can thus be classified as either right or left auxiliary trees, respectively, as determined by the location of the non-foot material. To maintain the invariant that textual material falls only on a single side of the spine, adjunction must be restricted so that left auxiliary trees may not adjoin into a node on the spine of a right auxiliary tree and vice versa. This prevents the formation of "wrapping" trees in which there are terminal symbols on both sides of the foot node. This restriction coupled with the requirement that all elementary auxiliary trees be non-wrapping is sufficient to limit the formalism to context-free expressivity and $O(n^3)$ parsability. In addition, Schabes and Waters (1993) demonstrate that TIG, like TAG, can lexicalize context-free grammars without changing the shape of the trees produced. For further background and discussion of TIGs and LTIGs, see the papers by Schabes and Waters (1993, 1995) and Hwa (2001).

Synchronous TIG (STIG) extends TIG just as SCFG extends CFG, by making elementary structures pairs of TIG trees with links between particular nodes in those trees. An STIG is a set of triples, $\langle t_L, t_R, \frown \rangle$ where t_L and t_R are elementary TIG trees and $\frown$ is a the linking relation between nodes in t_L and nodes in t_R (Shieber, 1994). Derivation proceeds as in TIG except that all operations must be paired. That is, a tree can only be substituted or adjoined at a node if its pair is simultaneously substituted or adjoined at a linked node. Figure 8 contains a sample English/French grammar fragment and Figure 9 shows the derivation of the paired sentences: "Jean likes red candies" and "Jean aime les bonbons rouges".

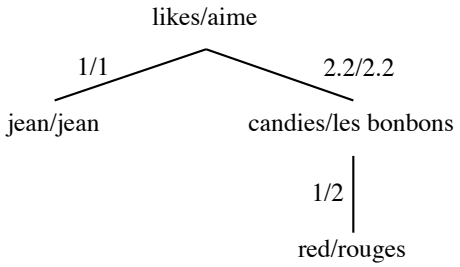


FIGURE 9. The derivation of "Jean really likes candies."/"Jean aime les bonbons rouges."

In support of our hypothesis about the utility of synchronous TIG in providing the properties desired in a synchronous grammar formalism for engineering translation systems, Hwa (2001) shows that a probabilistic version of TIG can have language modeling performance at the level of bigram models thereby capturing lexical relationships, while also retaining the advantages of context-free grammars in capturing syntactic structure. A synchronous TIG can easily express lexically-based dependencies, can be parsed in $O(n^6)$ time, and can handle both the substitution and adjunction requirements described above. Thus, a probabilistic version of synchronous TIG seems to possess all of the properties that we would like as the basis for a syntax-aware translation formalism.

In the succeeding sections, we develop parsing, parameter estimation, and translation algorithms for probabilistic synchronous TIGs. We follow the useful synchronous parsing framework laid out by Melamed (2004). Following Melamed, we distinguish multilingual *parsing*, in which a pair of sentences is analyzed as per a multilingual grammar, from *translation*, in which a single sentence is analyzed as per a multilingual grammar so as to determine all multilingual tuples in which the sentence is admitted. In addition, Melamed (2004) breaks generalized parsers into four distinct pieces, each of which may be analyzed separately: a grammar, a logic, a semiring, and a search strategy. The grammar is the language-specific set of rules and/or lexical items used by the parser. The logic is a set of inference rules that determine how the items of the grammar may be combined to form a derivation. The semiring determines what is computed by the parser as it parses; for instance, it might determine simple acceptability of a sentence by a grammar or it might determine the probability of the sentence being produced by the grammar. The search strategy refers to the order in which the inference rules are applied when more than one rule can apply. Melamed draws on previous work by Shieber, Schabes, and Pereira (1994) for his inference-rule-based separation of logics, grammars, and search strategy. He draws on previous work by Goodman (1999) for the method of parameterizing the parser by a semiring.

In this work, we present a statistical MT system within Melamed's framework, the heart of which is a synchronous parser for tree-insertion grammars. We use the parser in the context of an Expectation Maximization algorithm to iteratively parse a corpus of sentence-aligned bilingual data and estimate the parameters of

the synchronous grammar. The resulting grammar is used for machine translation of new sentences.

3. Parsing Synchronous Tree-Insertion Grammars

In this section we describe the necessary logic for parsing synchronous TIG. Little previous work has been done on multilingual parsing of (as opposed to translation by) synchronous formalisms. Melamed and Wang give logics for several variants of multitext grammar (Melamed, 2004). Other machine translation systems that can be characterized in the generalized parsing framework have also tackled this problem but with a less direct approach (Wu, 1996, 1997; Alshawi et al., 2000). Here we give an original parsing algorithm for a restricted subset of synchronous tree-insertion grammars. We also discuss the limitations of the algorithm, how it could be extended to support unrestricted synchronous TIG, and the resulting change in complexity.

3.1. **Parsing TIGs.**

TIGs are parsable in $O(n^3)$ time. Schabes and Waters (1995) give an Earley-style parsing algorithm for TIGs. Schabes and Waters (1993) give a CKY-style parsing algorithm for a slight variant, lexicalized context-free grammars (LCFG), which can straightforwardly be used to create a CKY-style parsing algorithm for TIGs.

For reasons to be discussed below, the choice of parsing algorithm has significant consequences. We have chosen to use CKY-style parsing because of the simplicity and clarity of the algorithm. Figure 10 presents inference rules for a CKY-style parsing algorithm for TIG. These rules can be used in conjunction with a standard chart parsing algorithm to parse an input sentence $w_1 \dots w_n$ according to a TIG. This algorithm is an adaptation of the algorithm given in Schabes and Waters (1993).

As all deductive parsing algorithms (Shieber et al., 1994), the algorithm works by generating items. Each item is of the form $[\eta, I, AdjSet]$, where η is a node in some elementary tree of the grammar,[3] I is an interval (i, j) between string positions i and j characterizing the substring $w_{i+1} \cdots w_j$ covered by the item, and $AdjSet$ is a subset of $\{L, R\}$ specifying which adjunctions, left or right, respectively, are still allowed at the node. We assume that each node η in an elementary tree is associated with a nonterminal or terminal label $Label(\eta)$ and with a set of possible adjunctions $Adj(\eta) \subseteq \{L, R\}$ in accordance with the TIG restrictions. Root nodes satisfy $Root(\eta)$ and if η_1 is the root of a tree that can substitute or left- or right-adjoin at node η_2, then $Subst(\eta_2, \eta_1)$, $Adjoin(\eta_2, \eta_1, L)$, and $Adjoin(\eta_2, \eta_1, R)$ are respectively satisfied. We write $\eta_0 \rightarrow \eta_1 \cdots \eta_k$ to indicate that η_0 dominates nodes $\eta_1, \dots, \eta_k$. (As is standard for CKY-style algorithms, we assume that all trees are at most binary branching.) We make use of an interval union operation $\cup_x$, parameterized by the order in which the intervals abut, where x is either L or R, defined by

$$\begin{array}{rcl} (i,j) \cup_L (j,k) & = & (i,k) \\ (j,k) \cup_R (i,j) & = & (i,k) \end{array} ,$$

and is otherwise undefined. Similar predicates and functions appear in later sets of rules following this same pattern; they are fully described in the appendix.

[3]The node may be thought of as specified by the name of the tree and the Gorn address of the node within the tree.

String to parse: $w_1 \cdots w_n$

Item Form: $[\eta, I, AdjSet]$

Goal Item: $[\eta, (0, n), \emptyset]$ $\quad Label(\eta) = StartSymbol,\ Root(\eta)$

Axioms:

WordAx $\quad \dfrac{}{[\eta, (i, i+1), \emptyset]} \quad Label(\eta) = w_{i+1}$

AuxAx $\quad \dfrac{}{[\eta, (i, i), \emptyset]} \quad Foot(\eta)$

Inference Rules:

SibCat $\quad \dfrac{[\eta_1, I, \emptyset]\ [\eta_2, J, \emptyset]}{[\eta_0, I \cup_L J, AdjSet(\eta_0)]} \quad \eta_0 \rightarrow \eta_1\ \eta_2$

SingPar $\quad \dfrac{[\eta_1, I, \emptyset]}{[\eta_0, I, AdjSet(\eta_0)]} \quad \eta_0 \rightarrow \eta_1$

Subst $\quad \dfrac{[\eta_1, I, \emptyset]}{[\eta_2, I, \emptyset]} \quad Root(\eta_1),\ Subst(\eta_2, \eta_1)$

Adjoin $\quad \dfrac{[\eta_1, I, AdjSet + x]\ [\eta_2, J, \emptyset]}{[\eta_1, I \cup_x J, AdjSet]} \quad Root(\eta_2),\ Adjoin(\eta_1, \eta_2, x)$

NoAdj $\quad \dfrac{[\eta, I, AdjSet + x]}{[\eta, I, AdjSet]}$

FIGURE 10. CKY-style inference rules for TIG parsing

Section 4 extends these rules for use with synchronous TIG (STIG), so we explain here what each rule does.

- The first axiom (WordAx) adds items to the chart for each node labeled with a word in the input sentence. These will be the anchors of trees that may be either initial or auxiliary. Since they are terminal nodes, no adjunctions are possible.
- The second axiom (AuxAx) adds items to the chart for the foot node of each auxiliary tree in the lexicon, making each auxiliary tree available for adjunction at each string position in the input sentence.
- The sibling concatenation (SibCat) and single parent (SingPar) rules simply move the derivation from child node to parent node in the tree. They only apply when all adjunction operations on the child node are completed, as evidenced by the empty $AdjSet$s in the antecedent items.
- The substitution rule (Subst) applies at the root node of initial trees whenever all adjunction operations are completed at that node. It creates new items for each node into which the completely processed initial tree could substitute, covering the span of the input string that the completed initial tree covers.
- The adjunction rule (Adjoin) applies whenever a node has an adjunction available in its adjunction set and there is an auxiliary tree that can adjoin at that node on the specified side without violating the TIG wrapping tree restrictions.

Strings to parse:	$\langle w_1 \cdots w_{n_S}, v_1 \cdots v_{n_T} \rangle$	
Item Form:	$\langle [\eta_S, I], [\eta_T, J], LinkSet \rangle$	
Goal Item:	$\langle [\eta_S, (0, n_S)], [\eta_T, (0, n_T)], \emptyset \rangle$	$Label(\eta_S) = srcStartSym$ $Label(\eta_T) = trgStartSym$ $RootPair(\overline{\eta})$
Axioms:		
WordAx	$\dfrac{}{\langle [\eta_S, (i, i+1)], [\eta_T, (l, l+1)], \emptyset \rangle}$	$w_{i+1} = Label(\eta_S)$ $v_{l+1} = Label(\eta_T)$
AuxAx	$\dfrac{}{\langle [\eta_S, (i, i)], [\eta_T, (l, l)], \emptyset \rangle}$	$Foot(\eta_S)$ $Foot(\eta_T)$
EmptyAx	$\dfrac{}{\langle [\eta_\epsilon, (i, i)], [\eta_\epsilon, (j, j)], \{(x, y)\} \rangle}$	$x, y \in \{L, R\}$ $EmptyTree(\eta_\epsilon)$
Inference Rules:		
SibCat	$\dfrac{\langle [\eta_{1S}, I_1], [\eta_{1T}, J_1], \emptyset \rangle \ \langle [\eta_{2S}, I_2], [\eta_{2T}, J_2], \emptyset \rangle}{\langle [\eta_S, I_1 \cup_L I_1], [\eta_T, J_1 \cup_x J_2], LS(\eta_S, \eta_T) \rangle}$	$\eta_S \rightarrow \eta_{1S}\ \eta_{2S}$ ($\eta_T \rightarrow \eta_{1T}\ \eta_{2T}$ and $x = L$ or $\eta_T \rightarrow \eta_{2T}\ \eta_{1T}$ and $x = R$)
SParSrc	$\dfrac{\langle [\eta_{1S}, i, j], [\eta_T, l, m], \emptyset \rangle}{\langle [\eta_S, i, j], [\eta_T, l, m], LS(\eta_S, \eta_T) \rangle}$	$\eta_S \rightarrow \eta_{1S}$ $NoLinks(\eta_{1S})$
SParTrg	$\dfrac{\langle [\eta_S, i, j], [\eta_{1T}, l, m], \emptyset \rangle}{\langle [\eta_S, i, j], [\eta_T, l, m], LS(\eta_S, \eta_T) \rangle}$	$\eta_T \rightarrow \eta_{1T}$ $NoLinks(\eta_{1T})$
Subst	$\dfrac{\langle [\eta_{1S}, i, j], [\eta_{1T}, l, m], \emptyset \rangle}{\langle [\eta_{2S}, i, j], [\eta_{2T}, l, m], \emptyset \rangle}$	$RootPair(\overline{\eta_1})$ $Subst(\overline{\eta_2}, \overline{\eta_1})$
Adjoin	$\dfrac{\langle [\eta_{1S}, I_1], [\eta_{1T}, J_1], LS + (x, y) \rangle \ \langle [\eta_{2S}, I_2], [\eta_{2T}, J_2], \emptyset \rangle}{\langle [\eta_{1S}, I_1 \cup_x I_2], [\eta_{1S}, J_1 \cup_y J_2], LS \rangle}$	$Adjoin(\overline{\eta_1}, \overline{\eta_2}, x, y)$ $RootPair(\overline{\eta_2})$

FIGURE 11. Inference Rules for CKY-style STIG parsing

- The no adjunction rule (NoAdj) simply skips the next available adjunction at a node without performing any adjunction operation.

3.2. **Parsing Synchronous TIG.** Our CKY-style STIG parsing algorithm is a straightforward generalization of the TIG parsing algorithm given in Figure 10. Since synchronized operations can only occur at linked nodes, we are able to do away with the adjunction sets. Instead, we keep track of the set of unused links between the two nodes in an item.[4] Each time one of the links is used, it is removed from the set. The inference rules for the CKY-style RSTIG parsing algorithm are shown in Figure 11.

[4]Note that because we do not allow empty anchors for trees and because every operation proceeds synchronously, the parsing complexity is actually only $O(n^4)$ because we can keep track of the starting index for the source and target spans, the length of the span, and a single guess as to where to break the span into sub-pieces. The complexity increases in the next section, when provision is made for parsing of sentence pairs of differing lengths.

We review the inference rules in Figure 11 in reference to the rules defined earlier in Figure 10. Note the simplification that comes from adding links. Rather than maintaining a set of available adjunctions at each node, that set is entirely determined by the links at that node. Given that each item is a pair of nodes, we do not need to consider all of the links at either node, but only the links between the two nodes in question. For convenience, we abbreviate node pair η_S and η_T from source and target trees, respectively, as $\overline{\eta}$. We use the notation $LS(\eta, \rho)$ to indicate the set of links between nodes η and ρ. We use the predicate $NoLinks(\eta)$ to signify that η has no links to any other node, including those not in the current item.

We are able to eliminate NoAdj rules entirely. Wherever there is no link at a particular node, no adjunction can take place, so that case is automatically handled. When we wish to skip a particular link, we simulate this by adjoining in a special auxiliary tree pair consisting of trees ($EmptyTree$) with a single node that is both root and foot. Thus, $EmptyTree$s do not change the shape of the derived tree. No additional inference rules are needed because the adjunction rules already handle this case. It does require the addition of an axiom to put the $EmptyTree$ nodes into the chart.[5] This change eliminates the tricky problem of estimating no-adjunction parameters.

3.3. **Completeness of CKY Parsing of Synchronous TIGs.** For the most part, the TIG parsing rules can be extended to the synchronous case simply by making the antecedent and consequent items into pairs of trees rather than single trees as is shown in the previous section. However, the restrictions imposed by the links make this generalization insufficient to handle certain cases. Intuitively, the problem arises when inference rule A's antecedents cannot be satisfied except by the consequent of inference rule B and vice versa. We explore this problem in detail below.

3.3.1. *The Crossing Problem.* Parsing TIGs requires only a straightforward generalization of well-understood parsing algorithms such as CKY parsing or Earley's algorithm. However, because these algorithms encode a particular procedural order for visiting the nodes in a tree to be parsed, the resulting parsers will accept proper, overlapping but distinct subsets of the STIG languages. That is, the obvious generalization of the CKY and Earley parsing algorithms described above are not complete for the STIG languages. The difficulty arises because the links between trees may "cross" in ways that make it impossible for the parsing algorithm to perform operations at the ends of both links. This is most easily illustrated by an example, such as the tree in Figure 12.

In CKY parsing, traversal of the trees proceeds from all the leaves simultaneously up to the root of the tree. When the parsing algorithm has finished processing all children of a node, it proceeds to the parent. Thus, in order to make use of the link between node B and node C (in Figure 12), the parsing algorithm must first finish

[5]The addition of $EmptyTree$s breaks lexicalization of an otherwise lexicalized grammar, but neither removes the linguistic advantages of lexicalization nor the parsing advantage that comes from not allowing adjunctions that don't increase the span of the item. The reason the latter is not a problem is that an empty tree can only adjoin to a link once because the link is then removed. Thus no spurious adjunctions are introduced. In our implementation we do not actually add the empty tree nodes to the chart but instead just make them available in every chart cell with no cover.

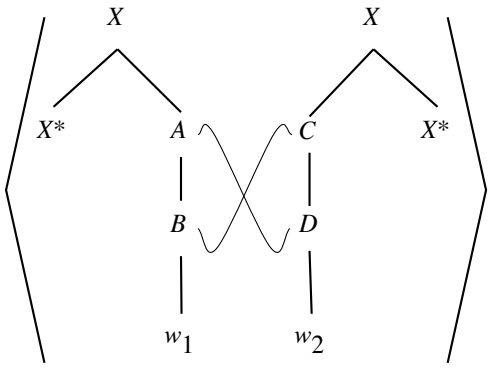


FIGURE 12. STIG links that pose a problem for CKY parsing

with node D. However, in order to make use of the link between node A and node D, the parser must first finish with node B. Thus, although a tree pair such as the one in Figure 12 is permitted by the STIG formalism, the parser will rule out any derivation that makes use of both of links in the tree pair.

In Earley's algorithm, the nodes are parsed in a top-down, left-first order; the left branches precede the right branches and the left sides of nodes precede the right sides of nodes. Thus, the trees in Figure 12 would be parsable using a version of Earley's algorithm for STIG, because the right side of node B and the left side of node C would both be reached before the right side of node A and the left side of node D. Earley's algorithm would be unable, however, to parse other combinations of links that do not pose a problem for CKY parsing.

3.4. **The Incomplete Cover Problem.** Because all operations in the grammar are synchronous, this problem can arise even with nodes that have no links between them. That is, at each point in the derivation, a node in a tree is paired with a node in the other tree in its tree pair. When two pairs of nodes cross each other across the boundaries of a branch in the tree, we have the same problem. This problem is illustrated in Figure 13. In order to perform a sibling concatenation to reach nodes Y_1 and Y_2, the derivation must have reached all of the daughter nodes of both and those daughter nodes must be paired with each other in some combination. If a pair of nodes, such as w_1 and w_2, are not both daughters or descendants of Y_1 and Y_2, the derivation will never be able to proceed. Note that although w_1 and w_2 are not explicitly linked, in order for them to be introduced into the derivation as anchors of their respective trees, they must have been introduced as a pair using the WORDAX inference rule. Note also that as a derivation proceeds, a node may be paired with different nodes because of the application of the SINGPAR inference rules, which are our only rules that apply asynchronously in any sense. Thus the problem persists even when the pairing is not of a daughter of Y_2 with an ancestor of Y_1 but also any descendant of Y_2 with an ancestor of Y_1.

To formalize the problematic situation, we introduce the notion of a cover. We say that set of nodes S covers an ancestor node X if all descendants of X that are

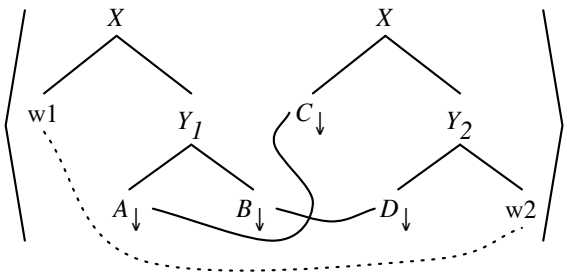


FIGURE 13. Paired node configurations that pose a problem for CKY parsing. Note that the dotted line is not a link, but just an illustration that w_1 and w_2 are paired nodes.

paired nodes are contained in S.[6] We prohibit tree structures in which the set of nodes that cover a node, here Y_1, are not paired with the set of nodes that cover Y_1's paired node Y_2.

The source of both this problem and the crossing problem is the limitation on the number of indices that we keep track of in a given item. If we allow an extension in which we keep a list of spans that an item covers, then we could extend the algorithm to do an operation on one half of the pair while keeping track of the obligation to do the corresponding operation when it becomes possible on the other half of the pair. However, limiting the algorithm to only two indices for each half of each item means that we only need to consider all sets of two spans and how to break them each into two pieces. This requires looping over six variables (corresponding to the ends of the spans and the dividing point we are considering in each span). Thus the algorithm requires $O(n^6)$ time. In the presentation of rules given before, the complexity is restricted further because the spans covered by an item must be of equal length on both the source and target side, so we can reduce to keeping track of only four independent quantities, making the algorithm require a worst case of $O(n^4)$. If we allow the algorithm to keep track of arbitrarily many spans within an item, the algorithm becomes exponential in n. Given the large increase in worst-case complexity, we restrict ourselves to the subset of TIG that does not exhibit these configurations of paired nodes. In the future we intend to try modifications to the algorithm if necessary, for instance allowing a single gap in each item.

3.4.1. *Definition of Restricted Synchronous TIG.* We define a restricted subset of STIG: restricted STIG (RSTIG), for which we assert that the given CKY parsing algorithm is complete. An RSTIG is a STIG where the linking relation in the elementary trees obeys the following rules:

- If a particular elementary tree pair contains a link between node A and node B, it may not also contain a link between: (a) an ancestor of A and a descendant of B, or (b) an ancestor of B and a descendant of A.

[6]Note that unpaired nodes, if they existed, would not present this problem. Although our rules currently do not allow unpaired nodes, we introduce modifications to allow them in Section 3.6.

- If a set of nodes form a cover of an ancestor, X, then the nodes with which those nodes are paired must form a cover of any nodes with which X may be paired.

The following definitions apply:

- **Paired.** Two nodes are paired if they may form an item in the course of a derivation.
- **Cover.** A set of nodes S covers an ancestor node X if all descendants of X that are paired nodes are contained in S.

3.5. **Allowing Translations of Differing Lengths.** As currently written the inference rules can only successfully recognize sentences in which there is a one-to-one correspondence between the anchors of paired trees in the lexicon, because the WordAx axiom always introduces a word in both the source and target derivations. This is an unacceptable constraint since sentences that are translations of each other will not in general have an exact one-to-one word correspondence. To remedy this problem, we can add trees explicitly anchored by the empty string, without losing many of the benefits we originally gained by requiring lexical anchors in the elementary tree pairs. Rather than requiring that each tree in each language have a non-empty anchor, we relax the restriction so that only one of the trees in any given pair of trees in the grammar must have a non-empty anchor.

Our inference rules do not require lexicalized tree pairs to work, so the only necessary modification to the rules is the addition of two axioms that permit tree pairs with an ϵ anchor to be entered into the chart:

$$\textsc{WordEpsAx} \quad \frac{}{\langle [\eta_S, (i, i+1)], [\eta_T, (l, l)], \emptyset \rangle} \quad \begin{array}{l} w_{i+1} = Label(\eta_S),\ \epsilon = Label(\eta_T) \\ Anchor(\eta_S, src, TP) \\ Anchor(\eta_T, trg, TP),\ TP \in G \end{array}$$

$$\textsc{EpsWordAx} \quad \frac{}{\langle [\eta_S, (i, i)], [\eta_T, (l, l+1)], \emptyset \rangle} \quad \begin{array}{l} v_{l+1} = Label(\eta_T),\ \epsilon = Label(\eta_S) \\ Anchor(\eta_S, src, TP) \\ Anchor(\eta_T, trg, TP),\ TP \in G \end{array}$$

This modification can as much as triple the size of the grammar, which will have a detrimental effect on the space and time complexity of the parsing algorithm as well as the space and time complexity of the expectation maximization algorithm based on it. However, because each operation that combines trees (excepting the adjunction of *EmptyTree*s discussed above) still increases the cover of the item in at least one of the sentences, the time complexity of the algorithm in terms of sentence length remains the same.

It also affects the time and space complexity of the translation process. The translator has only the source sentence as input and it must guess if there are any tree pairs in the parse that have a tree anchored with the empty string on the source side. Theoretically there can be arbitrarily many such tree pairs in the translation. We use a heuristic that limits the size of the target sentence relative to the source sentence to prevent the parser from searching a potentially very large set of possible translations.

The addition of ϵ-anchored trees on one side of tree pairs is sufficient to allow sentence pairs in which the length of the source and target sentence differ. Thus, it satisfies one of our desired properties for the formalism.

3.6. **Asynchronous Processing and Merge Rules.** In the previous section we modified the restrictions on the grammar to allow sentences of differing length. This models the situation in which a word in one language translates into nothing in the other language. Although this frequently occurs, it is not the only source of differing length in translated sentences. For instance, there may be idiomatic phrases in one language, such as "let the cat out of the bag" that tend to be translated to phrases of different length in the other, such as HCICD's "non è più un segreto". Although the grammar with only relaxed lexicalization may be sufficient to model this sort of complexity, we can also address the problem more directly.

In addition, the rules given thus far restrict the shape of trees to be close to isomorphic (modulo chains of single parent relations). Although we cannot allow arbitrary tree pair configurations without violating the restricted STIG definition, with rules to allow more asynchronous processing we can allow tree pairs in which the two trees have significantly different shapes.

We will first modify the grammar to allow an elementary tree to have more than one anchor. In addition, we allow trees to be paired with other trees with differing number of anchors. We must then modify the parsing rules to enable them to parse trees of this form. We can do this by addition of rules that process just one side of a tree pair up until the point where an operation takes place at a link in the tree pair or the tree pair is substituted or adjoined into another item.

This solution significantly expands the number of inference rules required. The new rules fall into four categories:

- Asynchronous Rules. These rules process only one side of a tree pair. Both antecedent and consequent items are asynchronous.
- Merge Rules. These rules take entirely asynchronous antecedents and produce a synchronous consequent item. These rules may only apply when a merge is necessary.
- Semi-Synchronous Rules. These rules apply when only one of the antecedent items is synchronous. This situation occurs when one of the antecedents of the rule has merged for an earlier synchronous operation. A merge is required when any operation combines other items with a synchronous item, whether that operation is synchronous or not.
- Synchronous Rules. These rules apply when all antecedents are synchronous. This includes all of the rules introduced in previous sections.

Due to the large number of merge and asynchronous rules, Figure 14 presents only a sampling of these rules: the axiom for creating an asynchronous source item, an asynchronous source sibling concatenation rule, the merge substitution rule, and a semi-synchronous adjunction rule. A complete listing of the asynchronous and merge inference rules is given in Appendix A, along with the appropriate calculations of semiring values to be discussed in Section 4.

The axiom in Figure 14 adds an asynchronous item for each lexical item with a source tree anchored by a word in the source sentence. If the lexical item can participate in a successful derivation it will later have to merge with its target half to create a synchronous item. Before merging, however, it may undergo sibling concatenation on the source side. This would occur if two asynchronous source items corresponded to sibling nodes in the same lexical item and the derivation did not require any substitution or adjunction operations on those nodes before moving on to process their parent node. If the derivation reaches the root of both a source

WordAxSrc	$\dfrac{}{\langle[\eta_S, (i, i+1)], -, \emptyset\rangle}$	$w_{i+1} = Label(\eta_S)$ $Anchor(\eta_S, src, TP)$
SibCatSrc	$\dfrac{\langle[\eta_{S1}, I_1], -, \emptyset\rangle \ \langle[\eta_{S2}, I_2], -, \emptyset\rangle}{\langle[\eta_S, I \cup_x J], -, \emptyset\rangle}$	($\eta_S \rightarrow \eta_{S1}\ \eta_{S2}$ and $x = L$ or $\eta_S \rightarrow \eta_{S2}\ \eta_{S1}$ and $x = R$) $NoLinks(\eta_{S1}),\ NoLinks(\eta_{S2})$
MrgSubst	$\dfrac{\langle[\eta_S, I], -, \emptyset\rangle \ \langle -, [\eta_T, J], \emptyset\rangle}{\langle[(\rho_S, I], [\rho_T, J], LS(\rho_S, \rho_T)\rangle}$	$Subst(\overline{\rho}, \overline{\eta}),\ RootPair(\overline{\eta})$ $NoLinks(\eta_S),\ NoLinks(\eta_T)$
SemiSyncAdj1	$\dfrac{\begin{array}{c}\langle[\eta_S, I_1], [\eta_T, J_1], LS + (x, y)\rangle \\ \langle[\rho_S, I_2], -, \emptyset\rangle \ \langle -, [\rho_T, J_2], \emptyset\rangle\end{array}}{\langle[\eta_S, I_1 \cup_x I_2], [\eta_T, J_1 \cup_y J_2], LS\rangle}$	$RootPair(\overline{\rho})$ $NoLinks(\rho_S),\ NoLinks(\rho_T)$ $Adjoin(\overline{\eta}, \overline{\rho}, x, y)$

FIGURE 14. Selected asynchronous and merge rules for CKY-style RSTIG parsing

item and a target item that happen to be part of the same lexical item, they will have to merge before substituting into another tree. The merge substitution rule performs this operation. Alternatively, it may participate in an adjunction with its asynchronous target counterpart and an already synchronous item, using the semi-synchronous adjunction rule shown.

Asynchronous and merge rules elegantly obviate the need for pairing words in one language with ϵ in the other. Rather than simulating lexical items made of multiple, possibly non-contiguous anchor words by pairing some anchors with ϵ, these rules allow us to write down these more complex, non-isomorphic tree pairs directly. One advantage of this change is that we can augment the tree pairs we learn with tree pairs that we harvest from another source, such as a treebank of synchronized tree pairs or a bilingual dictionary. As long as the trees obey the TIG restrictions and the links do not violate our restricted STIG constraints, they can simply be added into the set of tree pairs.

4. Parsing Probabilistic RSTIG

In order to use the RSTIG parser to induce an RSTIG grammar from data, we need to add probabilities to each item, corresponding to the inside probability of the source and target nodes in that item covering their respective parts of the source and target input sentences. Following the Melamed (2004) framework and the work of Goodman (1999) on parameterizing parsers with semirings, we are able to do this quite easily. In this section, we review the concept of a semiring and demonstrate how several useful semirings can be applied to the RSTIG inference rules and grammar.

4.1. Semirings.

Goodman (1999) demonstrates that the quantities most commonly computed by parsers can all be computed using the same parsing algorithm by simply swapping in a semiring that aids in calculating the particular quantity desired. The basic idea is that the parsing algorithm can be written to compute a quantity, such as acceptance or inside probability, by using a set of semiring operators in certain places. To change the quantity computed is to change the semiring being used.

A semiring contains two operators, a product ($\otimes$) and sum ($\oplus$), and two identities, 0 and 1. We require that $\oplus$ be associative and commutative and that $\otimes$ be associative and distribute over $\oplus$. 0 is an additive identity element and 1 is a multiplicative identity element. We will write $\langle S, \oplus, \otimes, 0, 1 \rangle$ for the semiring over set S with multiplicative operator $\otimes$ and identity 1 and additive operator $\oplus$ and identity 0.

In addition to the above, Goodman (1999) also includes a requirement that the semiring be complete. A complete semiring also allows for infinite sums that are associative and commutative and allows the multiplicative operator to distribute over infinite sums. Completeness is necessary to handle parsers in which there may be an infinite number of derivations for a particular item. This occurs when an item can derive itself, either directly or through a series of steps. Although all the semirings we describe here are complete, because operations can only take place at links and links are removed after an operation occurs, there will never be an infinite number of derivations of an item in our system.

To parameterize the parser by a semiring, each item and inference rule will have a semiring value associated with it. The semiring value associated with the consequent will be the product of the antecedent items' semiring values and the semiring value of the inference rule itself. When an item is added to the chart, if it is not present in the chart it will be added with this computed value. If it is already present in the chart, the new value will be its semiring value in the chart plus the value computed by the inference rule that just generated it. Note that if we use the Boolean semiring and give each inference rule the semiring value 1, we yield the same results as we did with the unparameterized parsing algorithm.

In addition to replicating the recognizer previously presented, we can also use semirings to compute inside probabilities and Viterbi derivations for the items in the chart. The inside probability of an item is the probability of that item being produced by the grammar starting at any point within the grammar. This probability is used in expectation maximization to induce a synchronous TIG grammar from data. The Inside semiring is defined as $\langle \mathbb{R}_0^\infty, +, \times, 0, 1 \rangle$. We set the semiring values associated with each inference rule as follows:

- The semiring value associated with all axiom rules will be 1. Since no decisions have been made in the parse at this point, each axiom item has probability 1 of being generated in the position in which it is inserted into the chart.
- The semiring values associated with sibling concatenation and the single parent rules will be 1. They are both entirely determined by the structure of the trees themselves. There is nothing probabilistic in their application.
- The semiring value associated with substitution of item B into item A will be the probability that B substitutes into A. Thus there will be a probability distribution over all items that can substitute into A. This probability will be given as part of the grammar and could be learned from a corpus.
- The semiring value associated with adjunction of item B into item A will be the probability that B adjoins into A. Again, there will be a probability distribution over all items that can adjoin into A given as part of the grammar or learned from a corpus. Note that we do not need to maintain a separate probability of no adjunction occurring because that probability

Semiring	S	$\oplus$	$\otimes$	0	1
Recognition	{FALSE, TRUE}	$\vee$	$\wedge$	FALSE	TRUE
Inside	$[0,1]$	$+$	$\times$	0	1
Viterbi	$[0,1]$	max	$\times$	0	1
Viterbi Derivation	$[0,1] \times \mathcal{D}$	$\max_{vit}$	$(\times, \cdot)$	$(0, \emptyset)$	$(1, \langle\rangle)$

FIGURE 15. Summary of the semirings used. The set of derivations $\mathcal{D}$.

is maintained as the probability of the appropriate *EmptyTree* within this distribution.

The Viterbi derivation is the derivation tree of the most probable parse for a given input. This value will be of use when we use our induced grammar to translate new sentences. The Viterbi derivation semiring is given in Figure 15 along with the others used in the translation system. The elements of the semiring are pairs consisting of the probability of the item and the derivation of the item. The additive operator takes the maximum probability item and its associated derivation. Although retaining only a single derivation when multiple derivations have the same probability destroys associativity, Goodman (1999) notes that in practice this is not a problem.[7] The multiplicative operator is just $\times$ for the probability and concatenation for the derivations. The additive identity is 0 for the probability and the empty set for the derivation. The multiplicative identity is 1 for the probability and the empty derivation. Concatenation with the empty derivation is an identity operation. The semiring values associated with the rules will use the same probabilities as with the Inside semiring. The pairs will be filled out with the empty derivation so that they will be identity elements when multiplied in.

4.2. **RSTIG Parsing Parameterized by a Semiring.** In this section, we present inference rules for parsing RSTIG parameterized by a semiring. The semiring values associated with the application of each inference rule are the parameters that must be learned from the corpus. Thus we present these parameters as well. Figure 16 contains a sampling of the inference rules modified to include the calculation of the semiring values associated with each item.

For each axiom item, the item is added with the multiplicative identity as its semiring value. For the sibling concatenation and single parent operations, because no decision is being made, the semiring values associated with those rules are always the multiplicative identity.

For substitution, the semiring value of the consequent is the result of multiplying the semiring value of the item being substituted by the semiring ring value of the substitution site by the semiring value of the substitution rule as applied to the item being substituted and the substitution site. The semiring value of the substitution site is the multiplicative identity because no decisions have been made with respect to it before this point.

For adjunction, the semiring value of the consequent is the result of multiplying together the semiring values of the two antecedent items and the semiring value of the adjunction rule applying to those two items.

[7]To maintain associativity, we just keep a forest of best derivations rather than a single best derivation.

Item Form: $\langle [\eta_S, I], [\eta_T, J], LinkSet, Value \rangle$

Goal Item:

GOAL $\quad \langle [\eta_S, (0, n_S)], [\eta_T, (0, n_T)], \emptyset, p \rangle \quad Label(\eta_S) = srcStartSym,\ Label(\eta_T) = trgStartSym,\ RootPair(\overline{\eta})$

Axioms:

WORDAX $\quad \dfrac{}{\langle [\eta_S, (i, i+1)], [\eta_T, (l, l+1)], \emptyset, 1 \rangle} \quad w_{i+1} = Label(\eta_S),\ v_{l+1} = Label(\eta_T)$

Inference Rules:

SIBCAT $\quad \dfrac{\langle [\eta_{S1}, I_1], [\eta_{T1}, I_2], \emptyset, p_1 \rangle \ \langle [\eta_{S2}, J_1], [\eta_{T2}, J_2], \emptyset, p_2 \rangle}{\langle [\eta_S, I_1 \cup_L I_2], [\eta_T, J_1 \cup_x J_2], LS(\eta_S, \eta_T), p_1 \otimes p_2 \rangle} \quad \eta_S \rightarrow \eta_{S1}\ \eta_{S2}$ ($\eta_T \rightarrow \eta_{T1}\ \eta_{T2}$ and $x = L$ or $\eta_T \rightarrow \eta_{T2}\ \eta_{T1}$ and $x = R$)

SUBST $\quad \dfrac{\langle [\eta_{S1}, I], [\eta_{T1}, J], \emptyset, p \rangle}{\langle [\eta_{S2}, I], [\eta_{T2}, J], \emptyset, p \otimes p_{Sub} \rangle} \quad RootPair(\overline{\eta_1}),\ Subst(\overline{\eta_1}, \overline{\eta_2}, p_{Sub})$

ADJOIN $\quad \dfrac{\langle [\eta_{S1}, I_1], [\eta_{T1}, J_1], LS + (x, y), p_1 \rangle \quad \langle [\eta_{S2}, I_2], [\eta_{T2}, J_2], \emptyset, p_2 \rangle}{\langle [\eta_{S1}, I_1 \cup_x I_2], [\eta_{T1}, J_1 \cup_y J_2], LS, p_1 \otimes p_2 \otimes p_{Adj}) \rangle} \quad RootPair(\overline{\eta_2}),\ Adjoin(\overline{\eta_1}, \overline{\eta_2}, x, y, p_{Adj})$

FIGURE 16. Selected inference rules for CKY-style RSTIG parsing parameterized by a semiring

The semiring value calculated for the string is then the sum of the values associated with the goal items times their probability of rooting the corresponding derivation:

$$\sum_{\langle [\eta_S, (0, srcLength)], [(\eta_T, (0, trgLength)], \emptyset, p \rangle} SP(\overline{\eta}) \otimes p \quad .$$

To make this more concrete, consider the application of the Inside semiring to the adjunction operation. The inside probability of the consequent item is the probability of the two items being adjoined together ($p_1 \otimes p_2$) multiplied by the probability (p_{Adj}, for instance) of those two items being adjoined together.

The semiring values associated with each rule are the parameters that will be estimated by the Expectation Maximization algorithm as described in Section 4.3. Since the application of the semiring makes explicit the parameters that will have to be estimated, these parameters are explained here. There are three categories of parameters to be estimated:

(1) **Adjunction Parameters.** Each adjunction link has an associated probability distribution over all of the tree pairs that could adjoin at that link. Since links specify on which side of the node the adjunction must take place, only a subset of the total tree pairs will be represented in any adjunction link's distribution. In addition, the nonterminal symbols at the nodes linked further restrict which tree pairs may adjoin.
(2) **Substitution Parameters.** Each substitution link has an associated probability distribution over all of the tree pairs that could substitute at

that link. The nonterminal symbols at the linked nodes restrict the tree pairs that will be represented in this distribution.

(3) **Start Tree Parameter.** This parameter specifies for each initial tree pair the of its serving as the root of a derivation, and is associated with the root nodes of the paired trees with the initial tree pairs of the grammar that are rooted in the start symbols of the grammars. This value is represented in the rules by a value for each pair of paired root nodes, $SP(\overline{\eta})$, that is multiplied into the semiring value of the goal item.

4.3. **Inducing a Synchronous Grammar from Sentence-Aligned Text.** The synchronous parser presented above comes to life when put in the context of an algorithm that can estimate the necessary parameters of the grammar. In this section we review the Inside-Outside algorithm, which is a special case of the Expectation Maximization algorithm used for estimating the parameters of a grammar (Prescher, 2001). We apply the algorithm to synchronous tree-insertion grammars in a straightforward generalization of the algorithm presented by Hwa (2001).

4.3.1. *The Inside-Outside Algorithm.* The Inside-Outside algorithm was proposed by Baker (1979) and refined by Lari and Young (1991) as a method for performing maximum likelihood estimation for probabilistic context-free grammars. The objective of the algorithm is to estimate a probability to associate with each rule of a grammar so that the grammar can then be used to produce the most likely parse of unseen sentences.

In an unambiguous grammar these probabilities can be estimated directly by parsing a training corpus and keeping track of the number of times each rule is used. These counts can then be normalized to produce a probability to associate with each rule of the grammar using the following formula (Jurafsky and Martin, 2000):

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{Count}(\alpha \rightarrow \gamma)} = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

However, grammars are rarely unambiguous and our initial grammar will certainly be ambiguous. Thus, these values cannot be computed directly. Instead we must keep track of all of the different parses for a given sentence and a weight for each of those parses. The Inside-Outside algorithm allows us to do just this. It begins with randomly initialized probabilities for each parameter (Initialization Step). It then uses these rules to parse a training corpus and calculate the expected frequency with which each rule is used (E Step). These expected frequencies are then used to compute new probability estimates for the rules (M Step). The process continues until the rule probabilities converge to a local maximum for the training corpus. A high-level description of the algorithm is given in Figure 17.

In the E step we calculate the inside and outside probabilities for each of the parameters of the grammar. In the case of PCFGs, the inside probabilities correspond to the probability that a given nonterminal in the grammar derives a given substring of the current input sentence. The outside probabilities correspond to the probability that the entire input sentence is derived with a given nonterminal in the grammar covering a given substring in the input sentence. These two quantities are depicted as the grey areas in Figure 18.

In the M step we use the inside and outside probabilities calculated in the E Step to update the parameter probabilities. When multiplied together, the inside and

Initialize the parameters of the grammar randomly
Repeat until convergence:
    E Step:
        Compute the Inside and Outside probabilities for the grammar
    M Step:
        Update the parameters to maximize the likelihood of the training data

FIGURE 17. High-Level View of the Inside-Outside Algorithm

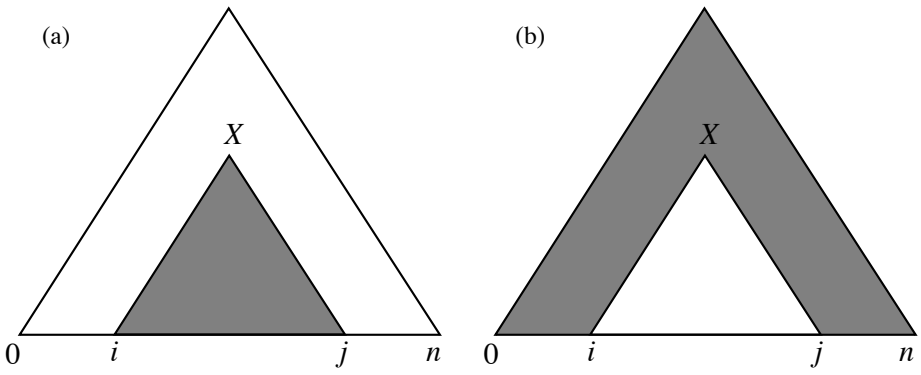


FIGURE 18. The (a) inside and (b) outside probability of the non-terminal symbol X deriving the substring from i to j

outside probabilities for a particular rule and sentence give the probability that the rule was used in the parse of the sentence. When divided by the probability of the sentence given all possible parses, we get the updated probability for the rule.

4.3.2. *The Inside-Outside Algorithm for Synchronous TIG.* The biggest decision to be made in adapting the Inside-Outside algorithm to synchronous TIG is to determine the tree pairs that make up the grammar and over which the probability distributions will be specified. We do not have a bank of tree pairs to draw on for our elementary trees, so we must construct them. In addition, because we are going to iteratively refine the parameters, we need to make sure that the initial grammar is sufficiently general to capture any pair of input sentences. In order to do this, we define a canonical tree form.

Following Hwa (2001), we note that the very simple tree pairs shown in Figure 19(a) allow us to simulate a bigram model. Expanding to capture some of the benefit of tree structure, the trees can be modified to have their anchors on the left as well as the right and to have an additional adjunction node between the root and the anchor where both right and left trees can adjoin. This is depicted in Figure 19(b). Note that the simple modification of allowing both left and right trees results in four different types of tree pairs. Finally, we add an additional adjunction node, again following Hwa's empirical findings that two adjunction sites were required in the monolingual TIG case to handle the variety and number of modifiers in natural language data (Hwa, 2001). This tree form is shown in Figure 19(c). Other canonical forms may be imagined as well. In order to ensure full

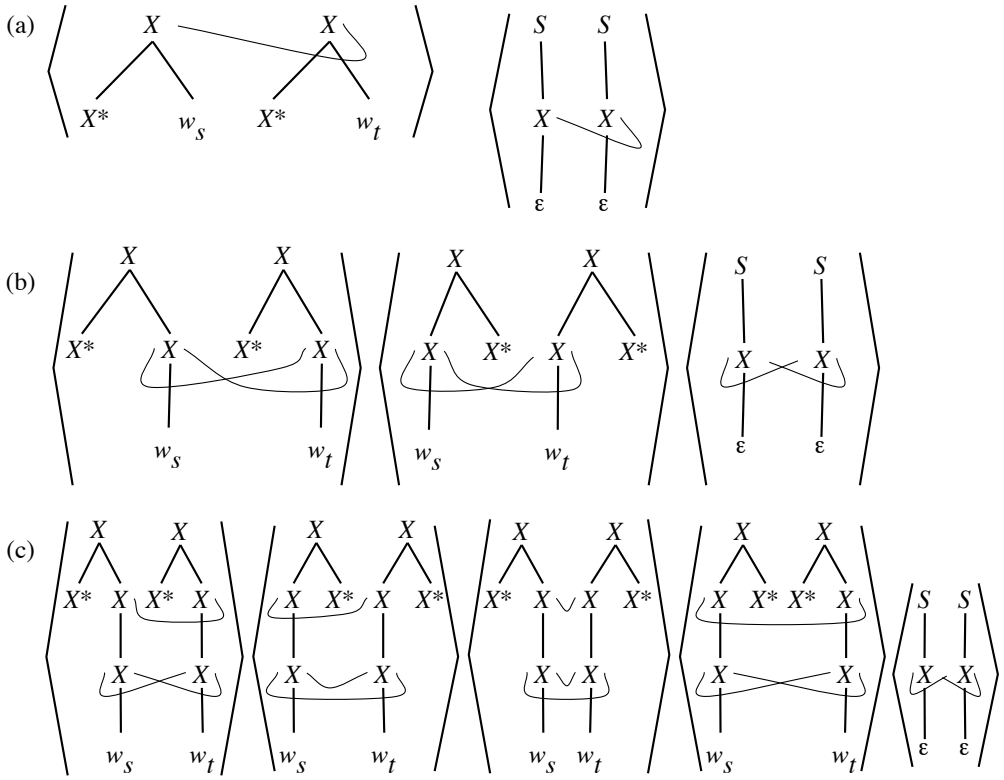


FIGURE 19. Possible canonical forms for synchronous TIG tree pairs: (a) the bigram model, (b) the extended bigram model, (c) our canonical tree pair form (showing two of the four auxiliary trees)

coverage of the input strings we generate tree pairs for each pair of source and target language words that appear in the source and target dictionaries.

In addition to the shape of the trees, we also have to specify the links. Since these links determine both where and how many adjunctions may take place, their placement is quite important. Different types of links help to express different dependencies. In particular, links between two nodes on the same side express dependencies where the order of the dependent with respect to the anchors are the same in the two languages. For instance, if adjectives precede nouns in both languages we would want LL links between noun nodes to express this dependency. Links between two nodes on different sides would correspondingly allow us to express dependencies where adjectives appear on different sides of the noun they modify in the two languages. Differences in the order of modifiers of the same lexical item can be expressed by links that change level in the tree. Links to nodes closer to the anchor will result in the dependent words appearing closer to the anchor. This difference could be used to express differences in the order of modifiers, for instance

different orderings for attachment of prepositional phrases. There are many possible combinations of links, even with our restrictions on link placement. In the experiments reported here, we use the links depicted in Figure 19(c). The canonical form trees could be augmented with hand-coded tree pairs to help the algorithm handle cases that prove to be difficult.

In the E step we parse all of the sentences in the training corpus while maintaining expected counts. The inside probabilities are calculated simply by parameterizing the parser with the Inside semiring as described above. The outside probabilities are then calculated in a pass over the chart, this time working from the root of the start trees down to the leaves. The intuitive idea is that we work backwards, undoing each operation that was done to parse each sentence. That is, we work from a consequent item back to its antecedent. The outside probability of an antecedent item is the product of the outside probability of the consequent item, the inside probability of the other antecedent item, and the semiring value of the operation being undone.

In the M step we update the probabilities of all of the parameters by normalizing the counts produced in the E step received for each parameter.

5. Preliminary Empirical Results

As an initial empirical test of the formalism and algorithms described here, we chose a simple, artificial language problem: translation of arithmetic expressions from postfix to infix, from `ABA+*` to `A*(B+A)` for instance.

The expression language includes constants `A` and `B` and operators `+` and `*`. This translation problem, though contrived, has several attractive properties as a simple test of syntax-aware MT techniques. First, it exhibits the type of hierarchical organization that we expect is difficult for non-syntax-aware translation systems. Second, the aligned corpora are easy to automatically generate, and translation correctness to automatically verify, even though, like natural language, there may be multiple correct target translations (differing in parenthesization) for a given source expression. Third, the need for extra parentheses in the target language forces differing lengths of source and target strings, so that this aspect of the translation formalism is exercised. Fourth, correctly aligned operators can be arbitrarily far apart in correct translations; in the terminology of IBM-style models, large distortions are manifest in translations, and these distortions are best characterized syntactically. Finally, the languages have very small vocabularies, which makes it possible to test the ability of our system in the absence of a fully optimized implementation.

We generated a corpus of aligned postfix and infix arithmetic expressions using a synchronous probabilistic expression-term-factor grammar that allows a single postfix expression to have several distinct infix translations that differ in the number and placement of parentheses, so long as sufficient parentheses are available to disambiguate properly according to the standard associativity and precedence of expressions. The inference rules used in the evaluation allow one tree in a tree pair to be anchored by ϵ, as in Section 3.5, but do not contain merge and asynchronous rules (Section 3.6). The full canonical form in Figure 19(c) was used, allowing (initially at least) any symbol in the source language to translate as any symbol in the target language, and, through pairing with ϵ anchors, any symbol in either langauge to be inserted freely.

Input expression	STIG Model Translation
B A B + B * +	B + (A + B) * B
A B A * + A *	(A + B * A) * A
A B B A + A * + +	A + B + ((B + A) * A)
B A A + A A + + + B *	(B + (A + A) + A + A) * B

FIGURE 20. Representative examples of correct translations of postfix arithmetic expressions to infix arithmetic expressions by our system.

Input expressions	Translation
A A A B + * + A A + *	(A * (A + B) + A) * A + A
A A B B * + +	A + A + (B * B
B A A * + B B + B * +	B + ((A + ((A)) * + B) * B

FIGURE 21. The three test expressions on which our MT system produced an incorrect result.

We trained the system on 411 aligned pairs of postfix and infix expressions with a maximum sentence length of 8 for the postfix expressions. We then tested the system on 90 test expressions without restriction on the length of the expressions, although most were not longer than 12 characters. The system produced correct infix expressions for 87 of the 90 test sentences, giving an error rate of 3.3%. Trained on 5000 pairs of aligned expressions, the system achieved 100% accuracy on the test set. Figure 20 shows representative examples of input expressions and their generated translations using the system trained on the small training set.

Of the three errors made by our system, two were relatively minor: in one of them a necessary set of parentheses around an addition is missing, the other is missing a right parenthesis. In the third erroneous translation it appears that the trained grammar was more seriously off track. Figure 21 shows the erroneous translations by our system.

As a baseline, we also trained GIZA++, a widely used implementation of an IBM-style system, on the same 411 training sentence pairs. The GIZA model trained on the small training set correctly translated 52 of 90 test sentences for an error rate of 42.2%. Unsurprisingly, GIZA performed quite poorly because of its lack of syntax awareness. To verify that the problem was not merely a lack of training data, we also trained GIZA on a training set consisting of 10,000 aligned sentence pairs to see if the additional training data would improve its performance. Again, the GIZA model trained on the large training set performed poorly, correctly translating only 53 of the 90 test sentences for an error rate of 41.1%. As expected, the GIZA models did well with the shorter, simpler sentences that did not require significant long-distance movement of any operators or parenthesization in the infix translation. Both models exhibited a tendency not to move constants or operators long distances and neither ever correctly parenthesized an infix expression. Figure 22 shows the output of GIZA on some sample expressions that our system translated correctly. Figure 23 shows the GIZA translations of the three sentences that our

Input expression	GIZA translations	
	small training set	large training set
B A B + B * +	B A + B * B)	B * (B + A + B
A B A A A + + B * + +	A + B * A * A	A * (A + A *
A B B A + A * + +	A + B B A * A)	A + A * (A + B +
B A A + A A + + + B *	B A + A A + A) * B	B + A + A) * B

FIGURE 22. Results of GIZA on the sample sentences correctly translated by our system as shown in Figure 20.

Input expression	GIZA translations	
	small training set	large training set
A A A B + * + A A + *	((A + B * A * A)	A + A + A + A * B *
A A B B * + +	(A * B + B	A + A + B * B
B A A * + B B + B * +	B A * A + B + B * B)	B * (A + B + B * B) *

FIGURE 23. Translations produced by GIZA on the three sentences on which the STIG model produced an incorrect result.

system translated incorrectly. In only one case did the GIZA system trained on 10,000 sentences produce a correct translation where our model erred.

Given the small size of the training set, the error rate for our system is quite low on the test data and demonstrates that our system appropriately learned to translate postfix arithmetic expressions to infix arithmetic expressions. Further, the large gap in quality and accuracy of translations between our system and the GIZA models shows that the capacity of our system to model hierarchical relationships with substitution does help our system to correctly translate sentences that are difficult for an IBM-style model.

The system used in these experiments used relatively aggressive thresholding. Even so, training took an hour four 411 sentence pairs. By contrast, GIZA took only seconds to train on the same data. Our system run without the thresholding and optimizations achieved a similar error rate to the optimized system but took approximately 30 hours to train. Since implementing the thresholding and other optimizations we have been able to train our system on 5000 pairs of arithmetic expressions. Training took approximately 12 hours. With this greater volume of training data our system achieved 100% accuracy on the test set.

6. Related Work

Melamed (2004, 2003) groups together syntax-aware machine translation efforts into a single framework based on generalized parsers as discussed above. His own work focuses on the use of multitext grammars, a generalization of context-free grammars to the multilingual case. He does not present an implementation of a machine translation system based on synchronous MG parsing. Thus, this work differs from his in that ours employs a different formalism for synchronous parsing

and presents a concrete implementation of an MT system based on synchronous parsing.

Hwa (2001) induces a tree-insertion grammar from monolingual data to learn a grammar for a single language. Her work inspired this work and provides much of the basis for our choice of the tree-insertion grammar formalism and our implementation of the Inside-Outside algorithm. This work can be seen as a generalization of her work to the multilingual case with machine translation as an additional resulting application. Hwa's later work on generating annotated treebanks for languages via syntactic projection across parallel texts is similar in character to this work, but has the different aim of generating annotated treebanks for languages that do not have them (Hwa, Resnik, Weinberg, Cabezas, and Kolak, 2002a) or evaluating when correspondences between syntactic trees can be drawn (Hwa, Resnik, Weinberg, and Kolak, 2002b).

Wu (1997, 1996) presents inversion transduction grammars as a base formalism for inducing a grammar over multilingual data. The normal form for inversion transduction grammar used in this work is equivalent to a multitext grammar in which rules are restricted to Chomsky Normal form. Thus, although the formalism is context-free equivalent, it is not lexicalized and cannot gain the advantages of relationships between lexical items. In addition, it cannot express discontinuous constituents. The finite-state head transducers of Alshawi et al. (2000) are similar and can express bilexical dependencies. However, they cannot express discontinuous constituents.

Eisner (2003) uses synchronous tree-substitution grammars to infer the correspondences between syntactic or dependency trees in two languages. Synchronous tree-substitution grammars lack the adjunction operation but are otherwise similar to TIGs. His method is analogous to the method used here, though the parsing algorithm differs significantly. In addition, the inputs to the method are fully formed trees. Thus only the correspondences between trees are learned, not the tree pairs themselves.

The work perhaps most similar in motivation and technique to that reported here is that of Chiang (2005) performing automated induction of synchronous context-free grammars. He too notes the advantage of synchronous grammars in principle over phrase-based finite-state translation, and in practice shows improved performance in Chinese-English translation as compared to a state-of-the-art phrase-based system.

Other work, such as that of Gildea (2003) and Yamada and Knight (2002), is similar in aims to our work but differs either in the inputs expected or the structure of the methods used. We do not review it here.

7. Conclusion

Today's machine translation systems miss many generalizations due to their inability to capture relationships between syntactic structures in the languages being translated. In this work we presented a system for statistical machine translation in the Melamed framework that infers syntactic structures for both the source and target languages and learns the correspondences between them. We motivated the choice of tree-insertion grammar because of its advantages in expressing bilexical relationships and handling discontinuous constituents and source and target sentences of differing lengths. We presented an $O(n^6)$ parsing algorithm for a restricted

subset of synchronous TIG along with a discussion of the restrictions as well as possible extensions. We concluded with a brief discussion of the incorporation of the parser, parameterized by various semirings, in a system for parameter estimation for the grammar.

Although empirical evaluation of the system beyond the successful proof of concept still remains, we believe synchronous TIG is a promising formalism for machine translation. In particular, it lends itself to simple extensions that allow the addition of hand-coded trees that can encourage the algorithm to capture generalizations it may miss when working with unlabeled data. The Penn Treebank of synchronous TAG elementary trees for various languages, many of which do not violate the TIG restrictions, provides a ready source for these trees, as do bilingual dictionaries.

The way forward in machine translation is to take advantage of syntactic information. This work is one step in that direction.

Acknowledgements

Partial support for the work reported in this paper was provided by the National Science Foundation under grant number IIS-0329089.

The authors would like to thank several people who aided in software development related to the research described in this paper. The code base for grammar induction for monolingual data was initially developed by Paul Govereau, Sasha Rush, and Daniel Mauer, based on code graciously provided by Rebecca Hwa. Additional aid and valuable consultation was provided by Rani Nelken and Chung-Chieh Shan.

References

Aho, Alfred V., and Jeffrey D. Ullman. 1969. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences* 3(1):37–56.

Alshawi, Hiyan, Srinivas Bangalore, and Shona Douglas. 2000. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics* 26(1):45–60.

Baker, J.K. 1979. Trainable grammars for speech recognition. In *Speech communication papers for the 97th meeting of the acoustical society of america*, ed. D.H. Klatt and J.J. Wolf, 547–550.

Brown, P.F., S.A. Della Pietra, V.J. Della Pietra, and R.L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics* 19(2):263–311.

Chiang, David. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (acl'05)*, 263–270. Ann Arbor, Michigan: Association for Computational Linguistics.

Eisner, Jason. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st annual meeting of the associate for computational linguistics*.

Gildea, Daniel. 2003. Loosely tree-based alignment for machine translation. In *Proceedings of the 41st annual meeting of the association for computational linguistics*, 80–87.

Goodman, Joshua. 1999. Semiring parsing. *Computational Linguistics* 25(4):573–605.

Hwa, Rebecca. 2001. Learning probabilistic lexicalized grammars for natural language processing. Ph.D. thesis, Harvard University.

Hwa, Rebecca, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2002a. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering* 1(1):1–15.

Hwa, Rebecca, Philip Resnik, Amy Weinberg, and Okan Kolak. 2002b. Evaluating translational correspondence using annotation projection. In *Proceedings of the 34th annual meeting of the association for computational linguistics*.

Joshi, Aravind K. 1985. *How much context-sensitivity is necessary for characterizing structural descriptions—tree-adjoining grammar*. Cambridge University Press.

Jurafsky, Daniel, and James H. Martin. 2000. *Speech and language processing*. Prentice-Hall.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT/NAACL*.

Lari, K., and S.J. Young. 1991. Applications of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language* 5:237–257.

Marcu, D., and W. Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the conference on empirical methods in natural language processing*.

Melamed, I. Dan. 2003. Multitext grammars and synchronous parsers. In *Proceedings of the 2003 human language technology conference of the north american chapter of the association for computational linguistics*, 79–86.

———. 2004. Statistical machine translation by parsing. Tech. Rep. 04–024, Proteus Project.

P. M. Lewis, II, and R. E. Stearns. 1968. Syntax-directed transduction. *Journal of the Association for Computing Machinery* 15(3):465–488.

Prescher, D. 2001. Inside-outside estimation meets dynamic EM. In *In proceedings of IWPT-2001*.

Schabes, Yves, Anne Abeille, and Aravind K. Joshi. 1988. Parsing strategies with 'lexicalized' grammars: Application to tree-adjoining grammars. In *Proceedings of the 12th conference on computational linguistics*, vol. 2, 578–583.

Schabes, Yves, and Richard C. Waters. 1993. Lexicalized context-free grammars. In *Proceedings of the 31st conference on association for computational linguistics*, 121–129. Association for Computational Linguistics.

———. 1995. Tree insertion grammar: A cubic time, parsable formalism that lexicalizes context-free grammars without changing the trees produced. *Computational Linguistics* 21(3):479–512.

Shieber, Stuart M. 1994. Restricting the weak-generative capacity of synchronous tree-adjoining grammars. *Computational Intelligence* 10(4):371–385.

Shieber, Stuart M., and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proceedings of the 13th international conference on computational linguistics*.

Shieber, Stuart M., Yves Schabes, and Fernando Pereira. 1994. Principles and implementation of deductive parsing. *Journal of Logic Programming* 24:503–512.

Wu, Dekai. 1996. A polynomial-time algorithm for statistical machine translation. In *Proceedings of the 34th annual meeting of the association for computational linguistics*.

———. 1997. Stochastic inversion transduction grammars and bilinguial parsing of parallel corpora. *Computational Linguistics* 23(3):377–404.

Yamada, Kenji, and Kevin Knight. 2002. A decoder for syntax-based statistical mt. In *ACL-02*.

Appendix A. Complete Asynchronous and Merge Inference Rules

Definition of Side Condition Predicates and Functions

$I \cup_x J$:	$(i,j) \cup_L (j,k) = (i,k)$ $(j,k) \cup_R (i,j) = (i,k)$ otherwise undefined
$LS(\eta_1, \eta_2)$:	The set of links between nodes η_1 and η_2
$RootPair(\overline{\eta})$:	True if nodes in node pair η are the roots of the trees in a pair
$NoLinks(\eta_1)$:	True if the given node has no links in its linkset with any other node
$Adjoin(\overline{\eta}, \overline{\rho}, x, y, p_{Adj})$:	True if tree pair rooted in the node pair $\overline{\rho}$ can adjoin into pair $\overline{\eta}$ using a link (x, y). Requires that (x, y) is in the linkset of $\overline{\eta}$, that nodes in node pair $\overline{\rho}$ root auxiliary trees of sidedness that matches (x, y), that the labels of $\overline{\eta}$ and $\overline{\rho}$ match, and that the adjunction not produce a wrapping tree. p_{Adj} is the semiring value of the operation.
$Subst(\overline{\eta}, \overline{\rho}, p_S)$:	True if tree pair rooted in the node pair $\overline{\rho}$ can substitute into pair $\overline{\eta}$. Requires that the nodes of $\overline{\eta}$ be linked for substitution and that the labels of $\overline{\eta}$ and $\overline{\rho}$ match. p_S is the semiring value of the operation.
$\eta \rightarrow \eta_1\ \eta_2$:	True if η is the parent of η_1 and η_2, in that order.
n_S, n_T:	The length of the input source or target sentence.
$SP(\overline{\eta})$	The semiring value associated with the use of the tree pair rooted at $\overline{\eta}$ to root a derivation.
$src/trgSent(i, j)$:	The words in the sentence between string positions i and j.
$Foot(\eta, side)$:	True if η is a foot node of a tree pair on the given side (src, trg).
$Anchor(\eta, side, TP)$:	True if the label at η is the anchor of a tree pair on the given side (src, trg).
$Label(\eta)$:	The label at node η.

Inference Rules

STRINGS TO PARSE $\langle w_1 \cdots w_{n_S}, v_1 \cdots v_{n_T} \rangle$

ITEM FORM $\langle [\eta_S, I], [\eta_T, J], LinkSet, Value \rangle$ where I, J of form (i, j)

GOAL ITEM $\langle [\eta_S, (0, n_S)], [\eta_T, (0, n_T)], \emptyset, Value \rangle$
where the value associated with the string pair is
$\sum_{\langle [\eta_S, (0, srcLength)], [(\eta_T, (0, trgLength)], \emptyset, p \rangle} SP(\overline{\eta}) \otimes p$

Asynchronous Rules

AXIOM-SRC
$$\frac{}{\langle [\eta_S, (i, i+1)], -, \emptyset, 1 \rangle}$$
$w_{i+1} = Label(\eta_S)$
$Anchor(\eta_S, src, TP)$

AXIOM-TRG
$$\frac{}{\langle -, [\eta_T, (l, l+1)], \emptyset, 1 \rangle}$$
$v_{l+1} = Label(\eta_T)$
$Anchor(\eta_T, trg, TP)$

AUX-AX-SRC
$$\frac{}{\langle [\eta_S, (i, i)], -, \emptyset, 1 \rangle}$$
$Foot(\eta_S, src)$

AUX-AX-TRG
$$\frac{}{\langle -, [\eta_T, (l, l)], \emptyset, 1 \rangle}$$
$Foot(\eta_T, trg)$

SP-SRC
$$\frac{\langle [\eta_{S1}, I], -, \emptyset, 1 \rangle}{\langle [\eta_S, I], -, \emptyset, 1 \rangle}$$
$\eta_S \rightarrow \eta_{S1}$
$NoLinks(\eta_{S1})$

SP-TRG
$$\frac{\langle -, [\eta_{T1}, J], \emptyset, 1 \rangle}{\langle -, [\eta_T, J], \emptyset, 1 \rangle}$$
$\eta_T \rightarrow \eta_{T1}$
$NoLinks(\eta_{T1})$

SC-SRC
$$\frac{\langle [\eta_{S1}, I_1], -, \emptyset, 1 \rangle \ \langle [\eta_{S2}, I_2], -, \emptyset, 1 \rangle}{\langle [\eta_S, I_1 \cup_L I_2], -, \emptyset, 1 \rangle}$$
$\eta_S \rightarrow \eta_{S1} \ \eta_{S2}$
$NoLinks(\eta_{S1}),\ NoLinks(\eta_{S2})$

SC-TRG
$$\frac{\langle -, [\eta_{T1}, J_1], \emptyset, 1 \rangle \ \langle -, [\eta_{T2}, J_2], \emptyset, 1 \rangle}{\langle -, [\eta_T, J_1 \cup_L J_2], \emptyset, 1 \rangle}$$
$\eta_S \rightarrow \eta_{T1} \ \eta_{T2}$
$NoLinks(\eta_{T1}),\ NoLinks(\eta_{T2})$

Merge Rules

MRG-ADJ
$$\frac{\begin{array}{c}\langle [\eta_S, I_1], -, \emptyset, 1 \rangle \ \langle -, [\eta_T, J_1], \emptyset, 1 \rangle \\ \langle [\rho_S, I_2], -, \emptyset, 1 \rangle \ \langle -, [\rho_T, J_2], \emptyset, 1 \rangle\end{array}}{\langle [\eta_S, I_1 \cup_x I_2], [\eta_T, J_1 \cup_y J_2], LS(\eta_S, \eta_T) - (x, y), p_{Adj} \rangle}$$
$RootPair(\overline{\rho})$
$NoLinks(\rho_S),\ NoLinks(\rho_T)$
$Adjoin(\overline{\eta}, \overline{\rho}, x, y, p_{Adj})$
$(x, y) \in LS(\eta_S, \eta_T)$

MRG-SUBST
$$\frac{\langle [\eta_S, I], -, \emptyset, 1 \rangle \ \langle -, [\eta_T, J], \emptyset, 1 \rangle}{\langle [(\rho_S, I], [\rho_T, J], \emptyset, p_S \rangle}$$
$Subst(\overline{\rho}, \overline{\eta}, p_S)$
$RootPair(\overline{\eta})$
$NoLinks(\eta_S),\ NoLinks(\eta_T)$

Semi-Synchronous Merge Rules

SP-SC
$$\frac{\langle [\eta_{S1}, I], [\eta_{T1}, J_1], \emptyset, p\rangle \ \langle -, [\eta_{T2}, J_2], \emptyset, 1\rangle}{\langle [\eta_S, I], [\eta_T, J_1 \cup_x J_2], LS(\eta_S, \eta_T), p\rangle}$$
$\eta_S \rightarrow \eta_{S1}$
($\eta_T \rightarrow \eta_{T1}\ \eta_{T2}$ and $x = L$ or
$\eta_T \rightarrow \eta_{T2}\ \eta_{T1}$ and $x = R$)

SC-SP
$$\frac{\langle [\eta_{S1}, I_1], [\eta_{T1}, J], \emptyset, p\rangle \ \langle [\eta_{S2}, I_2], -, \emptyset, 1\rangle}{\langle [\eta_S, I_1 \cup_x I_2], [\eta_T, J], LS(\eta_S, \eta_T), p\rangle}$$
($\eta_S \rightarrow \eta_{S1}\ \eta_{S2}$ and $x = L$ or
$\eta_S \rightarrow \eta_{S2}\ \eta_{S1}$ and $x = R$)
$\eta_T \rightarrow \eta_{T1}$

SSM-SC
$$\frac{\begin{array}{c}\langle [\eta_{S1}, I_1], [\eta_{T1}, J_1], \emptyset, p\rangle \\ \langle [\eta_{S2}, I_2], -, \emptyset, 1\rangle \ \langle -, [\eta_{T2}, J_2], \emptyset, 1\rangle\end{array}}{\langle [\eta_S, I_1 \cup_x I_2], [\eta_T, J_1 \cup_y J_2], LS(\eta_S, \eta_T), p\rangle}$$
($\eta_S \rightarrow \eta_{S1}\ \eta_{S2}$ and $x = L$ or
$\eta_S \rightarrow \eta_{S2}\ \eta_{S1}$ and $x = R$)
($\eta_T \rightarrow \eta_{T1}\ \eta_{T2}$ and $y = L$ or
$\eta_T \rightarrow \eta_{T2}\ \eta_{T1}$ and $y = R$)
$NoLinks(\eta_{S2}),\ NoLinks(\eta_{T2})$

SSM-ADJ1
$$\frac{\begin{array}{c}\langle [\eta_S, I_1], [\eta_T, J_1], LS + (x, y), p\rangle \\ \langle [\rho_S, I_2], -, \emptyset, 1\rangle \ \langle -, [\rho_T, J_2], \emptyset, 1\rangle\end{array}}{\langle [\eta_S, I_1 \cup_x I_2], [\eta_T, J_1 \cup_y J_2], LS, p \otimes p_{Adj}\rangle}$$
$RootPair(\rho_S, \rho_T)$
$NoLinks(\rho_S),\ NoLinks(\rho_T)$
$Adjoin(\eta, \rho, x, y, p_{Adj})$

SSM-ADJ-2
$$\frac{\begin{array}{c}\langle [\eta_S, I_1], -, \emptyset, 1\rangle \ \langle -, [\eta_T, J_1], \emptyset, 1\rangle \\ \langle [\rho_S, I_2], [\rho_T, J_2], \emptyset, p\rangle\end{array}}{\langle [\eta_S, I_1 \cup_x I_2], [\eta_T, J_1 \cup_y J_2], LS(\eta_S, \eta_T) - (x, y), p \otimes p_{Adj}\rangle}$$
$RootPair(\rho_S, \rho_T)$
$Adjoin(\eta, \rho, x, y, p_{Adj})$
$(x, y) \in LS(\eta_S, \eta_T)$

Synchronous Rules

SibCat
$$\frac{\langle [\eta_{S1}, I_1], [\eta_{T1}, J_1], \emptyset, p1\rangle \ \langle [\eta_{S2}, I_2], [\eta_{T2}, J_2], \emptyset, p2\rangle}{\langle [\eta_S, I_1 \cup_L I_2], [\eta_T, J_1 \cup_x J_2], LS(\eta_S, \eta_T), p1 \otimes p2\rangle}$$
$\eta_S \rightarrow \eta_{S1}\ \eta_{S2}$
($\eta_T \rightarrow \eta_{T1}\ \eta_{T2}$ and $x = L$ or
$\eta_T \rightarrow \eta_{T2}\ \eta_{T1}$ and $x = R$)

SPar-src
$$\frac{\langle [\eta_{1S}, I], [\eta_T, J], \emptyset, p\rangle}{\langle [\eta_S, I], [\eta_T, J], LS(\eta_S, \eta_T), p\rangle}$$
$\eta_S \rightarrow \eta_{S1}$
$NoLinks(\eta_{S1})$

SPar-trg
$$\frac{\langle [\eta_S, I], [\eta_{T1}, J], \emptyset, p\rangle}{\langle [\eta_S, I], [\eta_T, J], LS(\eta_S, \eta_T), p\rangle}$$
$\eta_T \rightarrow \eta_{T1}$
$NoLinks(\eta_{T1})$

Subst
$$\frac{\langle [\eta_S, I], [\eta_T, J], \emptyset, p1\rangle}{\langle [\rho_S, I], [\rho_T, J], \emptyset, p1 \otimes p2 \otimes p_S\rangle}$$
$RootPair(\eta_S, \eta_T)$
$Subst(\rho, \eta, p_S)$

Adjoin
$$\frac{\begin{array}{c}\langle [\eta_S, I_1], [\eta_T, J_1], LS + (x, y), p1\rangle \\ \langle [\rho_S, I_2], [\rho_T, J_2], \emptyset, p2\rangle\end{array}}{\langle [\eta_S, I_1 \cup_x I_2], [\eta_T, J_1 \cup_y J_2], LS, p1 \otimes p2 \otimes p_{Adj}\rangle}$$
$Adjoin(\eta, \rho, x, y, p_{Adj})$
$RootPair(\rho_S, \rho_T)$

Contents