Statistical Machine Translation by Parsing

I. Dan Melamed and Wei Wang
New York University

Designers of statistical machine translation (SMT) systems have begun trying to exploit tree-structured syntactic information. This article offers a coherent algorithmic framework to facilitate such efforts. Our main contribution is a generalization of the common notion of parsing. In an ordinary parser, the input is a single string, and the grammar ranges over strings. In order to use syntactic information, an SMT system requires generalizations of ordinary parsing algorithms that allow the input to consist of string tuples and/or the grammar to range over string tuples. Three particular generalizations, connected by some trivial glue, are all that is necessary for syntax-aware SMT:

- *A* **synchronous parser** *is an algorithm that can infer the syntactic structure of each component text in a multitext and simultaneously infer the correspondence relation between these structures.*
- *When a parser's input can have fewer dimensions than the parser's grammar, it is a* **translator**.
- *When a parser's grammar can have fewer dimensions than the parser's input, it is a* **synchronizer**.

This article offers a guided tour of these generalized parsing algorithms. It culminates with a recipe for using generalized parsing algorithms to train and apply a syntax-aware SMT system.

Introduction

Today's best statistical machine translation (SMT) systems can be viewed as weighted finite-state transducers (WFSTs) (Och and Ney, 2002; Kumar and Byrne, 2003). Such a WFST is typically composed of a series of transductions, as illustrated in Figure 1. Modeling translational equivalence using WFSTs is like approximating a high-order polynomial with line segments. The approximation can be arbitrarily good, given enough parameters. In practice, the number of parameters that can be reliably estimated is limited by the available training data and computing resources. Training data will always be limited for most of the world's languages. On the other hand, for resource-rich language pairs, where the amount of training data is practically infinite, the limiting factor is the number of model parameters that fit into our computers' memories. Either way, the relatively low expressive power of WFSTs limits the quality of SMT systems.

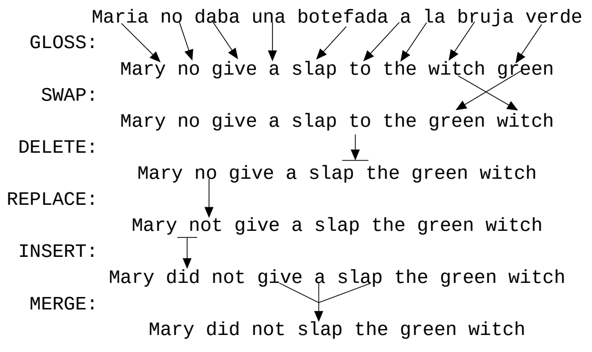
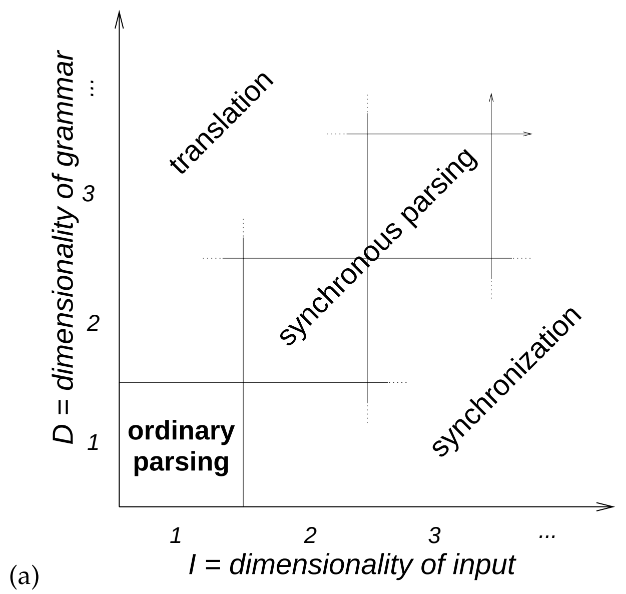

Maria no daba una botefada a la bruja verde
GLOSS:
Mary no give a slap to the witch green
SWAP:
Mary no give a slap to the green witch
DELETE:
Mary no give a slap the green witch
REPLACE:
Mary not give a slap the green witch
INSERT:
Mary did not give a slap the green witch
MERGE:
Mary did not slap the green witch

Figure 1

Translation by finite-state transduction. Adapted from Knight and Köhn (2003).



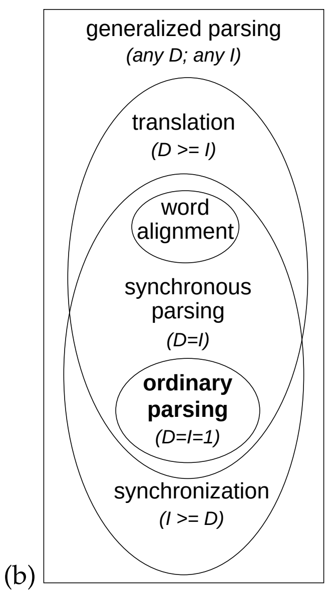


Figure 2
Two perspectives of how ordinary parsing algorithms can be generalized.

To advance the state of the art, SMT system designers have begun to experiment with tree-structured syntactic information (Wu, 1996; Alshawi, Bangalore, and Douglas, 2000; Yamada and Knight, 2002; Gildea, 2003). This article offers a coherent algorithmic framework to facilitate such efforts. The main contribution is a generalization of the common notion of parsing. A parser is an algorithm for inferring the structure of its input, based on a grammar that dictates what structures are possible or likely (Jurafsky and Martin, 2000). In an ordinary parser, the input is a single string, and the grammar ranges over strings. In order to use syntactic information, a machine translation system requires generalizations of ordinary parsing algorithms that allow the input to consist of string tuples and/or the grammar to range over string tuples. The kind of string tuples that are most relevant here are texts that are translations of each other, also called **parallel texts** or **multitexts**. Each multitext consists of **component texts** or **components**. The arity of the tuples is their **dimensionality**.

Figure 2 shows some of the ways in which ordinary parsing can be generalized. A **synchronous parser** is an algorithm that can infer the syntactic structure of each component text in a multitext and simultaneously infer the correspondence relation between these structures.[1] When a parser's input can have fewer dimensions than the parser's grammar, we call it a **translator**. When a parser's grammar can have fewer dimensions than the parser's input, we call it a **synchronizer**. The corresponding processes are called **synchronous parsing, translation** and **synchronization**, respectively. To our knowledge, synchronization has never been explored as a class of algorithms. Neither has the relationship between parsing and word alignment, which we discuss in Section 3. The relationship between translation and ordinary parsing was noted a long time ago (Aho and Ullman, 1969), but here we articulate it in more detail: ordinary parsing is a special case of synchronous parsing, which is a special case of translation.

To facilitate our exploration of generalized parsers, we shall decompose them into four parts: a grammar, a logic, a semiring, and a search strategy. The use of logics to describe parsers is not new , e.g., (Shieber, Schabes, and Pereira, 1995, and references therein). The parameterization of parsing algorithms by semirings was studied by Goodman (1998), among others. Klein and Manning (2003) have compared different search strategies for a fixed parsing logic and grammar. All of the generalizations of parsing in this article admit the same semirings and search strategies. Their differences lie in their grammars

[1] A suitable set of monolingual parsers can also infer the syntactic structure of each component, but cannot infer the correspondence relation between these structures.

and in their logics. Several previously published algorithms can also be viewed as generalized parsers (Aho and Ullman, 1969; Wu, 1996; Alshawi, Bangalore, and Douglas, 2000; Hwa et al., 2002). These other parsers are fundamentally similar to our parsers and to each other. The similarity is clearer when the semiring and search strategy are abstracted away.

This article uses multitext grammar as a running example of a grammar formalism that is a suitable foundation for syntax-aware statistical machine translation. The article begins with an informal introduction to multitext grammar, followed by a gentle introduction to synchronous parsing. It then offers a guided tour of the generalized parsing algorithms in Figure 2. It culminates with a recipe for using these algorithms to train and apply a syntax-aware SMT system.

1. A Simplistic Synchronous Parser

1.1 Binary-branching multitext grammars

Grammars that generate string tuples are often called **synchronous grammars**. WFSTs can be viewed as regular synchronous grammars. Multitext Grammars (MTG) are the natural generalization of context-free grammars (CFGs) to the synchronous case. MTGs can express arbitrary synchronous trees over arbitrarily many parallel texts. Every MTG is a D-MTG for some integer constant $D > 0$, and it generates multitexts with D components. Thus, a 1-MTG is a CFG; 2-MTGs generate bitexts.

An MTG has disjoint sets of terminals T and nonterminals N. We often group terminals or nonterminals into vectors that we call **links**. Links express the translational equivalence between their components. In MTG applications, the different components of a terminal link will often come from largely disjoint subsets of T, representing the vocabularies of different languages. Every link generated by a D-MTG has D components, some of which may be empty. An empty component indicates that an expression vanishes in translation. The special symbol $()$ denotes an empty link component.

Each MTG also has a set of **production rules** (or just **productions** for short). A production in 2-MTG might look like this:

$$\begin{array}{ccc} X & \multirow{2}{*}{$\Rightarrow$} & A^1 \; e \; C^3 \\ Y & & A^2 \; D^1 \; f \end{array} \tag{1}$$

There is one row per production component, on both the left-hand side (LHS) and the right-hand side (RHS). Each symbol on the LHS is either a nonterminal symbol or the dummy symbol $()$. Whenever a component on the LHS is $()$, the corresponding component on the RHS must also be $()$. A production component is **inactive** if it has $()$'s on both sides; otherwise it is **active**. The RHS of an active component is a string of terminals and/or indexed nonterminals. The indexes express translational equivalence: All the nonterminals with the same index constitute a link. The same nonterminal may appear in different components, either with the same index or with a different index (like A). Some nonterminals on the RHS might have no translation in some components, in which case there will be no co-indexed nonterminal in those components (as for A^2 or C^3). Terminals never have indexes. Correspondence information is not observable in terminal string tuples, just as it is not observable in raw multitext.

For simplicity, we shall limit our attention to MTGs in Generalized Chomsky Normal Form (GCNF) (Melamed, Satta, and Wellington, 2004). This normal form allows simpler algorithm descriptions than the normal forms used by Wu (1997) and Melamed (2003). In GCNF, every MTG production is either a terminal production or a nonterminal production. **Terminal productions** have the form

$$\begin{array}{ccc} \vdots & & \vdots \\ () & & () \\ X_i & \Rightarrow & t_1 \\ () & & () \\ \vdots & & \vdots \end{array} \tag{2}$$

Table 1
A 2-MTG with the production rules in (a) can derive the multitree in Figure 3 as shown in (b). Production 9 is not used in the derivation.

(a)

$$\begin{array}{r} S \\ S \end{array} \Rightarrow \begin{array}{ll} V^1 & NP^2 \\ NP^2 & V^1 \end{array} \tag{4}$$

$$\begin{array}{r} V \\ V \end{array} \Rightarrow \begin{array}{c} WASH^1 \\ MIT^2 \end{array} \tag{5}$$

$$\begin{array}{r} NP \\ NP \end{array} \Rightarrow \begin{array}{ll} D^1 & N^2 \\ & N^2 \end{array} \tag{6}$$

$$\begin{array}{r} N \\ N \end{array} \Rightarrow \begin{array}{c} DISH^1 \\ PAS^2 \end{array} \tag{7}$$

$$\begin{array}{r} WASH \\ () \end{array} \Rightarrow \begin{array}{c} \text{wash} \\ () \end{array} \tag{8}$$

$$\begin{array}{r} WASH \\ () \end{array} \Rightarrow \begin{array}{c} \text{clean} \\ () \end{array} \tag{9}$$

$$\begin{array}{r} () \\ MIT \end{array} \Rightarrow \begin{array}{c} () \\ \text{moy} \end{array} \tag{10}$$

$$\begin{array}{r} D \\ () \end{array} \Rightarrow \begin{array}{c} \text{the} \\ () \end{array} \tag{11}$$

$$\begin{array}{r} DISH \\ () \end{array} \Rightarrow \begin{array}{c} \text{dishes} \\ () \end{array} \tag{12}$$

$$\begin{array}{r} () \\ PAS \end{array} \Rightarrow \begin{array}{c} () \\ \text{Pasudu} \end{array} \tag{13}$$

(b)

$$\begin{array}{r} S \\ S \end{array} \rightarrow \begin{array}{ll} V^1 & NP^2 \\ NP^2 & V^1 \end{array}$$

$$\rightarrow \begin{array}{ll} WASH^3 & NP^2 \\ NP^2 & MIT^4 \end{array}$$

$$\rightarrow \begin{array}{lll} WASH^3 & D^5 & N^6 \\ N^6 & MIT^4 & \end{array}$$

$$\rightarrow \begin{array}{lll} WASH^3 & D^5 & DISH^7 \\ PAS^8 & MIT^4 & \end{array}$$

$$\rightarrow \begin{array}{lll} \text{wash} & D^5 & DISH^7 \\ PAS^8 & MIT^4 & \end{array}$$

$$\rightarrow \begin{array}{lll} \text{wash} & D^5 & DISH^7 \\ PAS^8 & \text{moy} & \end{array}$$

$$\rightarrow \begin{array}{lll} \text{wash} & \text{the} & DISH^7 \\ PAS^8 & \text{moy} & \end{array}$$

$$\rightarrow \begin{array}{lll} \text{wash} & \text{the} & \text{dishes} \\ PAS^8 & \text{moy} & \end{array}$$

$$\rightarrow \begin{array}{lll} \text{wash} & \text{the} & \text{dishes} \\ \text{Pasudu} & \text{moy} & \end{array}$$

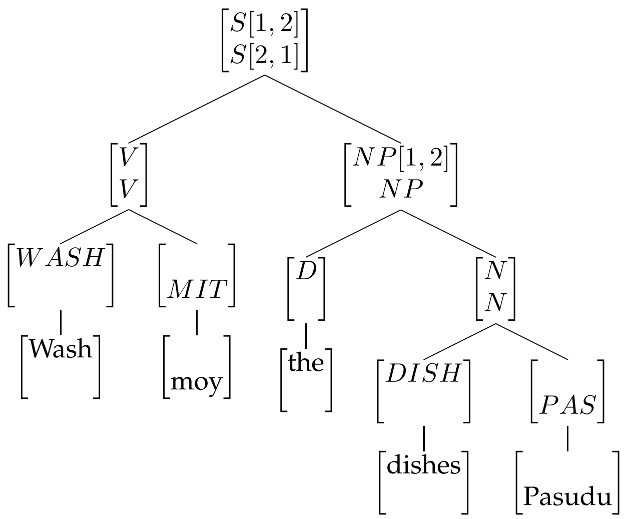
All components except for one are inactive. The active component has a nonterminal on the LHS and a terminal on the RHS. **Nonterminal productions** have the form

$$\begin{array}{c} X_1 \\ \vdots \\ X_D \end{array} \Rightarrow \begin{array}{c} \alpha_1 \\ \vdots \\ \alpha_D \end{array} \tag{3}$$

The LHS is a link of D nonterminals and/or ()s. In the active components, the α on the RHS is a string of one or more indexed nonterminals. In GCNF, every nonterminal production must have exactly two distinct indexes among all the αs on the RHS.

The D-MTG derivation process begins with the **start link**, which is a vector of D copies of the special start symbol $S \in N$. The derivation continues with nondeterministic application of production rules. The semantics of $\Rightarrow$ are the usual semantics of rewriting systems, i.e., that the expression on the LHS can be rewritten as the expression on the RHS. However, every nonterminal on the LHS of the production must match a nonterminal in the corresponding component of the derivation up to that point. Then, all the nonterminals with the same index are rewritten simultaneously. Every time the derivation process generates an instance of a nonterminal link, the link's components are co-indexed with a fresh index, to avoid confusion with previously derived instances of the same nonterminals. In this manner, D-MTGs generate D-tuples

(a)


$\begin{bmatrix}S[1,2]\\S[2,1]\end{bmatrix}$
$\begin{bmatrix}V\\V\end{bmatrix}$
$\begin{bmatrix}NP[1,2]\\NP\end{bmatrix}$
$\begin{bmatrix}WASH\\\end{bmatrix}$
$\begin{bmatrix}\\MIT\end{bmatrix}$
$\begin{bmatrix}D\\\end{bmatrix}$
$\begin{bmatrix}N\\N\end{bmatrix}$
Wash
moy
the
$\begin{bmatrix}DISH\\\end{bmatrix}$
$\begin{bmatrix}\\PAS\end{bmatrix}$
dishes
Pasudu

(b)

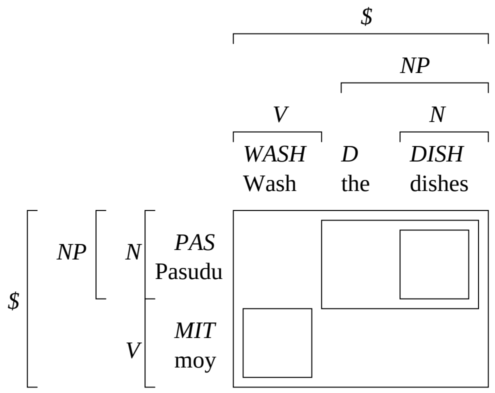


Figure 3
A 2D multitree in English and transliterated Russian. The two representations are equivalent: (a) Every internal node is annotated with the linear order of its children, in every component where there are two children. (b) Rectangles are 2D constituents.

of trees that are pairwise isomorphic up to reordering of sibling nodes and deletion of subtrees. Each tuple of such trees can also be viewed as one D-dimensional tree, as illustrated in Figure 3. We shall refer to multidimensional trees as **multitrees**.

The MTG derivation process can be represented by a derivation tree, just like the derivation process of CFGs. As for CFG, MTG derivation trees are identical to the resulting parse trees. For example, consider an MTG with the production rules in Table 1(a). That MTG can derive the multitree in Figure 3 as shown in Table 1(b).

The **rank** of an MTG production is the number of nonterminal links on its RHS, i.e. the number of unique indexes. Thus, Production 1 has rank 3, and terminal productions have rank zero. The **rank** of an MTG is the maximum rank of its production rules. MTG(R) is the class of MTGs of rank R. The MTG whose productions are in Table 1 is a 2-MTG(2). When the rank of an MTG is greater than 2, its nonterminals can be discontinuous, but we defer discussion of this issue until Section 2. In the next subsection, we restrict our attention to 2-MTG(2), which is equivalent to Inversion Transduction Grammar (Wu, 1997). Grammars whose derivation trees are always binary, such as MTG(2), are sometimes called **binary-branching** grammars.

1.2 Logic R2D2C

A parser's logic describes its possible states and actions. The state of a parser consists of a grammar and a set of item[2] instances. The set of production rules of the grammar can be viewed as the set of axioms of the logic; it does not change. The set of item instances grows monotonically, starting from the empty set. The parser terminates if and when its state includes the Goal item.

Inference rules describe the parser's possible actions. Each action is a transition from one state to another. We shall express inference rules as sequents: $\frac{Y_1,\ldots,Y_k}{X}$ means that the **consequent** X can be deduced from the **antecedents** $Y_1, \ldots, Y_k$. In other words, if a parser's state includes $Y_1, \ldots, Y_k$ then it can transition to a state that also includes X. Consequents are always items. An item that appears in an inference rule stands for the proposition that the item is part of the parser's state. In less abstract terms, this means that the item is in the parse chart. Each antecedent refers either to an item or to a grammatical constraint.

For expository purposes, we begin with Logic R2D2C, shown in Table 2. This is a logic for parsing the subclass of MTG restricted to 2-dimensional productions of rank 2, called 2-MTG(2). Logic R2D2C is simpler than the parsing logics used by Wu (1997) and Melamed (2003), because it uses only one type of item and it never composes terminals. Each of the logic's items contains up to two pairs of boundaries (i_1, j_1 and i_2, j_2) and a vector of two labels(X_1 and X_2). Span boundaries range over positions between and around the words in a text. The position to the left of the first word is zero, and the position to the right of the jth word is j. The item at the top of Table 2 represents a constituent between word boundaries $i_1 \ldots j_1$ of the first component and $i_2 \ldots j_2$ of the second component, labeled by the link inside the rectangle. Since MTGs can generate multitexts with components of unequal length, the height of an item need not equal the width. In particular, one of the dimensions can have zero width, in which case the boundary variables on that dimension are dummies and the label is (). Such one-dimensional items are necessary for representing multitree branches that are active in only one component.

Parser R2D2C is any inference procedure based on Logic R2D2C. Parser R2D2C begins with an empty chart. The only inferences that can fire in this state are those with no antecedent items (though they can have antecedent grammar terms). A *Scan* inference can fire for each input word that appears on the RHS of a terminal production in the grammar. In less abstract terms, this means that the parser's chart is initialized with all the words in the input. $G_T(p)$ is the value that the grammar assigns to the terminal production p. The range of this value depends on the semiring used (see below). A *Scan* inference can fire for the ith word $w_{d,i}$ in component d if that word appears in the dth component of the RHS of a terminal production in the grammar. If a word appears on the RHS of the relevant component of multiple productions (with different LHSs), then multiple *Scan* inferences can fire for that word. The active span of each item inferred by a *Scan* inference always has the form $(i-1, i)$ because such items always span one word, so the distance

[2]The term **item** refers to any partial parse.

Table 2
Logic R2D2C

Item Form	$\begin{array}{r} \\ i_2 \\ \\ j_2 \end{array} \overset{i_1 \quad j_1}{\boxed{\begin{matrix} X_1 \\ X_2 \end{matrix}}}$
Inference Rules *Scan* component 1	$\dfrac{\mathrm{G}_T \begin{pmatrix} X & \Rightarrow & w_{1,i} \\ () & & () \end{pmatrix}}{\begin{array}{r} \\ - \\ \\ - \end{array} \overset{i-1 \quad i}{\boxed{\begin{matrix} X \\ () \end{matrix}}}}$
Scan component 2	$\dfrac{\mathrm{G}_T \begin{pmatrix} () & \Rightarrow & () \\ X & & w_{2,i} \end{pmatrix}}{\begin{array}{r} \\ i-1 \\ \\ i \end{array} \overset{- \quad -}{\boxed{\begin{matrix} () \\ X \end{matrix}}}}$
Compose (1 pattern shown out of 8)	$\dfrac{\begin{array}{r} \\ k_2 \\ \\ j_2 \end{array} \overset{i_1 \quad k_1}{\boxed{\begin{matrix} Z_1 \\ Z_2 \end{matrix}}} , \begin{array}{r} \\ i_2 \\ \\ k_2 \end{array} \overset{k_1 \quad j_1}{\boxed{\begin{matrix} Y_1 \\ Y_2 \end{matrix}}} , \mathrm{G}_N \begin{pmatrix} X_1 & \Rightarrow & Z_1^1 \; Y_1^2 \\ X_2 & & Y_2^2 \; Z_2^1 \end{pmatrix}}{\begin{array}{r} \\ i_2 \\ \\ j_2 \end{array} \overset{i_1 \quad j_1}{\boxed{\begin{matrix} X_1 \\ X_2 \end{matrix}}}}$
Goal	$\begin{array}{r} \\ 0 \\ \\ n_2 \end{array} \overset{0 \quad n_1}{\boxed{\begin{matrix} S \\ S \end{matrix}}}$

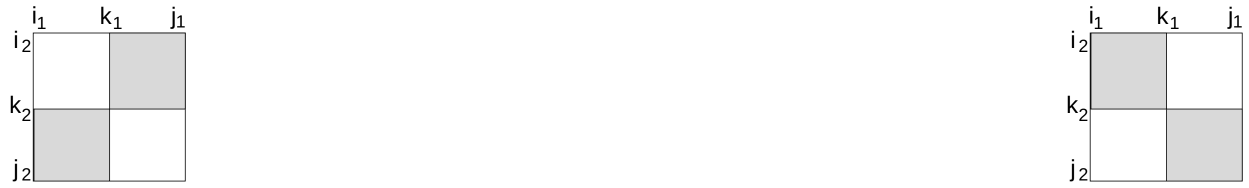


Figure 4
Parser R2D2C has two ways to compose two 2D items represented by the shaded regions into one item represented by their minimum enclosing rectangle. The pattern on the right corresponds to the *Compose* rule shown in Table 2.

between the item's boundaries is always one. Except in the active component, the labels and spans of *Scan* consequents are always empty.

Parser R2D2C spends most of its time composing pairs of items into larger items. A *Compose* inference can have two 2D antecedents, two 1D antecedents, or one of each. The case of two 1D antecedents is analogous to the *Link* inference rule in Melamed (2003)'s Parser R2D2A. The composition of a 2D item with a 1D item asserts that the yield of the 1D item vanishes in translation. The parser can compose two items whenever they satisfy both of the following constraints:

- *Immediate Dominance (ID).* Their label vectors match the two links on the RHS of a nonterminal production rule in the grammar.
- *Linear Precedence (LP).* The relative order of the two items in each dimension of the input must match their order in that production rule.

Both constraints are enforced by the G_N term of the *Compose* inference rule. $G_N(p)$ is the value that the grammar assigns to the nonterminal production p.

Under a binary-branching MTG, the LP constraints imply that the items must be adjacent without overlapping in every dimension where they are both active. In particular, if neither item has inactive components, then they must be corner-to-corner. Figure 4 illustrates the two ways that two 2D items can satisfy linear precedence under 2-MTG(2).

The LP constraints ensure that the final multitree covers each input word exactly once. An ordinary one-dimensional parser succeeds only if it infers a constituent that covers the input text. Analogously, a synchronous parser succeeds only if it infers a constituent that covers the multitext space spanned by the input multitext. The Goal item of Logic R2D2C represents such a constituent. An example of its instantiation is the outermost rectangle in Figure 3(b). Parser R2D2C succeeds on a bitext with component lengths n_1 and n_2 if and only if it deduces the item spanning the $n_1 \times n_2$ multitext space, labeled by the start link. The parser fails if it fires all possible inference rules without deducing the Goal item.

1.3 Semirings

A semiring consists of a set, two binary operators over that set, and an identity element in the set for each operator. A parser uses its semiring's two operators $\oplus$ and $\otimes$ to compute the value $V(\alpha)$ of an inference consequent α from the values of its antecedents $\alpha_1, \ldots, \alpha_k$:

$$V(\alpha) = \bigoplus_{\alpha_1,\ldots,\alpha_k,\ s.t.\ \frac{\alpha_1\ ,\ \cdots\ ,\ \alpha_k}{\alpha}} \bigotimes_{i=1}^{k} V(\alpha_i) \tag{14}$$

The α_i above can be any term — either an item or a grammar term such as G_T or G_N. Some examples will help to explain the above formula.

First, consider the boolean semiring over the set {TRUE, FALSE}, where $\oplus$ is boolean disjunction and $\otimes$ is boolean conjunction. Under this semiring, a grammar term is TRUE iff its production rule is in the grammar. Items are TRUE iff they are derivable: Equation 14 says that an item α is TRUE iff α is the consequent of some inference rule where all the antecedents are TRUE. Starting from the parser's initial state, we can run the parser under a Boolean semiring to determine whether the Goal item is TRUE. If so, then the parser succeeds on the input multitext, which means that the grammar can generate that multitext.

If the grammar guiding the parser is probabilistic, then it's possible to use an inside-probability semiring, where $\oplus$ is real addition and $\otimes$ is real multiplication. Under this semiring, the G terms return the probabilities of production rules. We can run the parser under the inside-probability semiring to compute the total probability of any item, including the Goal item. The probability of the Goal item is the probability of the grammar generating the input multitext.

Goodman (1998) studied the above semirings and a variety of other semirings that are useful for parsing, including:

- the Viterbi semiring, which can compute the probability of the single most probable derivation;
- the Viterbi-derivation semiring, which can compute the single most probable derivation;[3]
- the Viterbi-n-best semiring, which can compute the n most probable derivations;
- the derivation-forest semiring, which can compute all the possible ways of deriving the goal item;
- the counting semiring, which can compute the number of possible ways to derive the goal item.

All of these parsing semirings apply equally well to synchronous parsing, and to the other classes of algorithms that we discuss in this article.

More recently, Eisner (2002) has claimed that parsing under an expectation semiring is equivalent to the Inside-Outside algorithm for PCFGs. If so, then there is a straightforward generalization for Probabilistic MTGs (PMTGs) involving synchronous parsing. Parameter estimation for probabilistic synchronous grammars is an important application of synchronous parsers. However, parameter estimation is beyond the scope of this article.

1.4 Search strategies

The logic in Table 2 imposes only a partial order on its inferences, in that an inference cannot fire until other inferences deduce its antecedents. Beyond that, the logic is nondeterministic. A deterministic **search strategy** imposes a complete ordering on the possible inference rule instantiations. We can choose any complete order that is consistent with the partial order imposed by the logic. For example, we can decide to parse strictly bottom-up, inferring smaller items before larger ones. Alternatively, given weights (such as probabilities) for parse items, we can run the parser as a uniform-cost search, inferring less costly consequents before more costly ones. If we are interested in just the single best or the n-best ways of deriving the Goal, then A* strategies of varying sophistication can be employed to speed up the search (Klein and Manning, 2003).

2. A Parser for Arbitrary MTGs

Productions with more than two nonterminal links on the RHS are useful for representing subcategorization frames with more than two dependents. For inferring such productions, it is always more efficient to binarize an MTG than to allow a parser to compose more than two items at a time. However, binarization of MTG nonterminal productions can produce nonterminals with gaps. For every dimensionality $D \geq 2$ and rank $R \geq 4$, there are some correspondence patterns that can be generated by D-MTG(R), but not by D-MTG($R-1$) (Rambow and Satta, 1999). For example, a grammar in 2-MTG(4) can generate the multitree in Figure 5, but no grammar in 2-MTG(3) can generate it. The distinguishing characteristic of such multitrees is that no two constituents are adjacent in more than one dimension. Each set of sibling constituents in such an arrangement must be encapsulated in the RHS of a single nonterminal production. If the grammar is bilexical, then even productions of rank 3 can require discontinuous constituents for binarization (Melamed, 2003). This case can arise, for example, when two prepositional phrases switch places in translation. Synchronous grammars that do not allow discontinuous constituents, such as 2-MTG(2), are unlikely to have good coverage, even for multitexts involving languages that are syntactically similar.

[3] Or the set of most probable derivations, if there are ties.

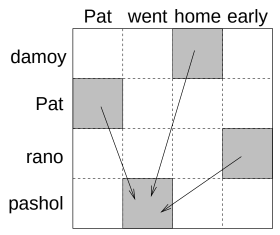


Figure 5
Discontinuous constituents are practically unavoidable in synchronous parsing, as for this bitext in English and (transliterated) Russian. Arrows are 2D dependencies. Shaded squares are 2D constituents.

Parser R2D2C is limited to inferring multitrees generated by 2-MTG(2). To deal with the full expressive power of MTG, this parser requires two generalizations. First, it must be able to handle more than two dimensions. Second, it must admit discontinuous items. Before describing this more general parser, we develop the prerequisite notation and computational machinery.

2.1 A more flexible notation

The production rule notation described in Section 1.1 uses superscripts to superimpose information about translational equivalence on top of information about the linear order of constituents. In order to describe more general parsers more compactly, we introduce an alternative notation for nonterminal productions, which facilitates independent reference to each kind of information. In the new notation, nonterminal links are written in columns, and their precedence within each component is indicated separately. The special symbol () acts as a placeholder for inactive link components. For example, Production 1 in the original notation becomes

$$\begin{matrix} X \\ Y \end{matrix} \Rightarrow \bowtie \begin{bmatrix} [1,2,4] \\ [3,1,5] \end{bmatrix} \begin{pmatrix} \mathrm{A} & \mathrm{e} & () & \mathrm{C} & () \\ \mathrm{D} & () & \mathrm{A} & () & \mathrm{f} \end{pmatrix} \tag{15}$$

in the new notation. The first index in the second component above is 3, and it refers to the third column in the link vector. Thus, we know that A comes first in that component. The list of link indexes in each component is a **precedence array**. E.g., the precedence array in the second component above is $[3,1,5]$. The elements of a precedence array never refer to links that are inactive in its component. All the precedence arrays in a given production rule constitute a **precedence array vector (PAV)**.[4]

Precedence arrays can express discontinuities. They can also indicate how to arrange parts of discontinuous subconstituents. For example, suppose that the first component of Production 15 had a gap between the e and the C. Suppose further that the D in the second component was expected to have a gap. Then the production might be written like this:

$$\begin{matrix} X(2) \\ Y \end{matrix} \Rightarrow \bowtie \begin{bmatrix} [1,2;4] \\ [1,3,1,5] \end{bmatrix} \begin{pmatrix} \mathrm{A} & \mathrm{e} & () & \mathrm{C} & () \\ \mathrm{D(2)} & () & \mathrm{A} & () & \mathrm{f} \end{pmatrix} \tag{16}$$

In the first component, the semicolon in the precedence array indicates the position of the gap. In the second component, the precedence array indicates that the two parts of D (the nonterminal in the first link) should wrap around the A (the nonterminal in the third link). Precedence arrays are more general than permutations, because precedence arrays can refer to the same position more than once.

The $\bowtie$ ("join") operator rearranges the symbols in each component according to that component's precedence array. For example,

$$\bowtie \begin{bmatrix} [2,3,4] \\ [1,3,2,5] \\ [4,3,1] \end{bmatrix} \begin{pmatrix} () & \mathrm{A} & \mathrm{B} & \mathrm{C} & () \\ \mathrm{W} & \mathrm{Y} & \mathrm{X} & () & \mathrm{Z} \\ \mathrm{T} & () & \mathrm{U} & \mathrm{V} & () \end{pmatrix} = \begin{matrix} \mathrm{A}^2\mathrm{B}^3\mathrm{C}^4 \\ \mathrm{W}^1\mathrm{X}^3\mathrm{Y}^2\mathrm{Z}^5 \\ \mathrm{V}^4\mathrm{U}^1\mathrm{T}^3 \end{matrix} \tag{17}$$

[4]PAVs are more informative than the RTVs used by Melamed (2003), because RTVs obscure link information.

For each production in the original CFG-style notation, there are many ways to re-express it in the new notation. The existence of multiple ways to express the same constraint is called spurious ambiguity, and it leads to wasted effort during parsing. To avoid spurious ambiguity, we stipulate a normal form for production rules in the new notation. The normal form requires that, if the arrays in the PAV are concatenated, then the first appearance of an index i must precede the first appearance of an index j for all $i < j$, except where the arrangement is incompatible with an earlier choice of indexes. We could, for example, obtain the same result above if we put $()Z()$ first, put $()WT$ last, and switch their indexes in the 2nd and 3rd precedence arrays. However, the normal form requires the 2nd precedence array to be [1, 3, 2, 5], not [5, 3, 2, 1], so $()Z()$ must be listed last in the link vector.

2.2 Discontinuous Links and Discountinuous Spans

In Production 16, X and D are annotated with their cardinalities. The **cardinality** of a constituent or nonterminal is the number of its contiguous spans. The process that transforms an MTG into GCNF adds cardinality information to nonterminal labels in production rules (Melamed, 2003).[5] We adopt the convention that a nonterminal label without cardinality information denotes a cardinality of 1. Links of possibly discontinuous nonterminals are called **discontinuous links** or **d-links**.

Now we introduce some notation for describing discontinuities in parse items, and some machinery for operating on them. Expanding on Johnson (1985), we define a **discontinuous span** (or **d-span**, for short) as a list of zero or more intervals $(l_1, r_1; \ldots; l_m, r_m)$, where

- the l_i are left span boundaries and the r_i are right span boundaries, so that $l_i < r_i$;
- $r_i \leq l_{i+1}$, which means that the intervals do not overlap;
- a d-span is in normal form if all the inequalities between r_i and l_{i+1} are strict; i.e., there is a gap between each pair of consecutive intervals;
- an empty d-span is denoted by $()$.

As in ordinary spans, d-span boundaries range over positions between and around the words in a text. Parse items have one d-span per dimension. We shall denote d-spans and vectors of d-spans by σ and τ. A d-span that pertains to only one particular dimension d is denoted with a subscript, as in σ_d. When a label or a d-span variable has both a superscript and a subscript, it refers to a range of dimensions; e.g., σ_j^i is a vector of d-spans, one for each dimension from i to j.

We define two operators over d-spans.

- $+$ is the concatenation operator: Given a set of d-spans, it outputs the union of their intervals in normal form. E.g., $(1, 3; 8, 9) + (7, 8) = (1, 3; 7, 9)$.
- $\wr$ is the relativization operator[6]: Given a sequence of d-spans, it computes the precedence array that describes the contiguity and relative positions of their intervals. E.g., $(1, 3; 8, 9) \wr (7, 8) = [1; 2, 1]$, because if these two d-spans were concatenated, then the result would consist of the 1st interval of the 1st d-span, followed by a gap, followed by the 1st interval of the 2nd d-span, followed by the 2nd interval of the 1st d-span.

The inputs of $+$ and $\wr$ must have no overlapping intervals, or else the output is undefined. Both operators apply componentwise to vectors of d-spans.

2.3 Logic C

Table 3 contains Logic C, which is a generalization of Logic R2D2C to arbitrary MTGs in GCNF. Parser C is any parser based on Logic C. The input to Parser C is a tuple of D parallel texts, with lengths $n_1, \ldots, n_D$.

[5] This notation is an abbreviation of the notation introduced by Melamed, Satta, and Wellington (2004). E.g., we write $X(3)$ instead of (X, X, X).

[6] Melamed (2003) used the $\otimes$ symbol for this operator, but we rename it here to avoid confusion with this symbol's traditional use in describing semirings.

Table 3
Logic C

Item Form	$\left[X_D^1; \sigma_D^1\right]$
Inference Rules	
Scan component d, $1 \leq d \leq D$	$\dfrac{\mathrm{G}_T\begin{pmatrix} ()_{d-1}^1 & & ()_{d-1}^1 \\ X & \Rightarrow & w_{d,i} \\ ()_D^{d+1} & & ()_D^{d+1} \end{pmatrix}}{\begin{bmatrix} ()_{d-1}^1 & & ()_{d-1}^1 \\ X & ; & (i-1,i) \\ ()_D^{d+1} & & ()_D^{d+1} \end{bmatrix}}$
Compose	$\dfrac{\left[Y_D^1;\tau_D^1\right] \ , \ \left[Z_D^1;\sigma_D^1\right] \ , \ G_N(X_D^1;\tau_D^1 \wr \sigma_D^1, Y_D^1, Z_D^1)}{[X_D^1;\tau_D^1+\sigma_D^1]}$
Goal	$\left[S_D^1; (0, n_d)_D^1\right]$

Since D can be more than 2, we must dispense with the graphical metaphor in Table 2. We also remove the $\Rightarrow$ from the G terms and merely list the relevant parameters[7], separating the LHS parameters from the RHS parameters by a semicolon. The parameters of G_T are the nonterminal link on the LHS, and the terminal link on the RHS. The parameters of G_N are the nonterminal link on the LHS, and the PAV and the two nonterminal links on the RHS.

Logic C's items consist of a D-dimensional label vector X_D^1 and a D-dimensional d-span vector σ_D^1. The items need d-spans, rather than ordinary spans, because Parser C needs to know all the boundaries of each item, not just the outermost ones. Some (but not all) dimensions of an item can be inactive, having an empty d-span and the label (). The notation $(0, n_d)_D^1$ indicates that the goal item spans the input from the left of the first word to the right of the last word in each component $d, 1 \leq d \leq D$. Thus, as always, the Goal item must be contiguous in all dimensions.

Parser C begins by firing *Scan* inferences, just like Parser R2D2C, but it can *Scan* from each of the D input components. The parser can also *Compose* pairs of items into larger items. The conditions for item composition are the ID and LP constraints described in Section 1.2, generalized to d-links of arbitrary dimensionality. These constraints are expressed using the d-span operators defined in Section 2.2: Parser C can compose two items if their labels are the d-links of a production rule in the grammar, and if the contiguity and relative order of their intervals is consistent with the PAV of that production rule.

3. Word Alignment

The class of synchronous parsers includes some algorithms for word alignment. A translation lexicon (weighted or not) can be viewed as a degenerate MTG (not in GCNF) where every production has the form

$$\begin{matrix} S & & t_1 \\ \vdots & \Rightarrow & \vdots \\ S & & t_D \end{matrix} \tag{18}$$

[7]This notation will facilitate the generalization in Section 5.

I.e., each production rewrites the start link into one terminal per component. Under such an MTG, the logic of word alignment is the one in Melamed (2003)'s Parser A, with one modification: Instead of a single Goal item, the Goal of word alignment is any *set* of items that covers the input exactly once. Also note that, since nonterminals do not appear on the RHS of production rules, *Compose* inferences are impossible and unnecessary, so they can be removed from the logic if desired. We shall refer to this logic as Logic W. This logic can be used with the expectation semiring (Eisner, 2002) to find the maximum likelihood estimates of the parameters of a word-to-word translation model.

4. Translation

A D-MTG can guide a synchronous parser to infer the hidden structure of a D-component multitext. Now suppose that we have a D-MTG and an input multitext with only I components, $I < D$. When some of the component texts are missing, we can ask the parser to infer a D-dimensional multitree that includes the missing components, which are supplied by the grammar. The resulting multitree will cover the I **input components/dimensions** among its D dimensions. It will also express the $D - I$ **output components/dimensions**, along with their syntactic structures. When a parser's input can have fewer dimensions than the parser's grammar, we call it a **translator**.

4.1 Translator CT

Table 4 shows Logic CT, which is a generalization of Logic C. Translator CT is any parser based on Logic CT. The items of Translator CT have a D-dimensional label vector, as usual. However, their d-span vectors are only I-dimensional. Recall that the purpose of d-spans is to enforce LP constraints, so that the input is covered exactly once. The only LP constraint that needs to be enforced for output dimensions is cardinality, not absolute position. The cardinality constraint is enforced by the role templates ρ_D^{I+1} in the G_N term. This mechanism also guarantees that the goal item will be contiguous in all dimensions, since the start link is never discontinuous.

Translator CT scans only the input components. Terminal productions with active output components are simply loaded, and their LHSs are added to the chart without d-span information. Composition proceeds as before, except that there are no constraints on the precedence arrays in the output dimensions — the precedence arrays in ρ_D^{I+1} are free variables.

In summary, Logic CT differs from Logic C as follows:

- Items store no position information (d-spans) for the output components.
- For the output components, the *Scan* inferences are replaced by *Load* inferences, which are not constrained by the input.
- The *Compose* inference does not constrain the d-spans of the output components. (Though it still constrains their cardinality.)

We have constructed a translator from a synchronous parser merely by relaxing some constraints on the output dimensions. Table 4 is so similar to Table 3 because Parser C is just Translator CT for the special case where $I = D$. The relationship between the two classes of algorithms is easier to see from their declarative logics than it would be from their procedural pseudocode or equations.

As in Parser C, the first few *Compose* inferences fired by Translator CT typically link items that have no dimensions in common. If one of the items exists only in the input dimension(s), and the other only in the output dimension(s), then this inference is, de facto, translation. The possible translations are determined by consulting the grammar. Thus, in addition to its usual function of evaluating syntactic structures, the grammar simultaneously functions as a translation model.

4.2 Translation semirings

Logic CT can be coupled with any of the semirings listed in Section 1.3. For example, under a boolean semiring, this logic will succeed on an I-dimensional input if and only if it can infer a D-dimensional

Table 4
Logic CT

Item Form	$\left[X_D^1; \sigma_I^1\right]$
Inference Rules	
Scan component d, $1 \leq d \leq I$	$\dfrac{\mathrm{G}_T\begin{pmatrix} ()_{d-1}^1 & & ()_{d-1}^1 \\ X & & w_{d,i} \\ ()_I^{d+1} & ; & ()_I^{d+1} \\ ()_D^{I+1} & & ()_D^{I+1} \end{pmatrix}}{\begin{bmatrix} ()_{d-1}^1 & & ()_{d-1}^1 \\ X & & (i-1,i) \\ ()_I^{d+1} & ; & ()_I^{d+1} \\ ()_D^{I+1} & & \end{bmatrix}}$
Load component d, $I < d \leq D$	$\dfrac{\mathrm{G}_T\begin{pmatrix} ()_I^1 & & ()_I^1 \\ ()_{d-1}^{I+1} & ; & ()_{d-1}^{I+1} \\ X & & t \\ ()_D^{d+1} & & ()_D^{d+1} \end{pmatrix}}{\begin{bmatrix} ()_I^1 & & ()_I^1 \\ ()_{d-1}^{I+1} & ; & \\ X & & \\ ()_D^{d+1} & & \end{bmatrix}}$
Compose	$\dfrac{\left[Y_D^1;\tau_I^1\right] \,,\, \left[Z_D^1;\sigma_I^1\right] \,,\, G_N\left(X_D^1 \;;\; \begin{matrix}\tau_I^1 \wr \sigma_I^1 \\ \rho_D^{I+1}\end{matrix} \,,\, Y_D^1 \,,\, Z_D^1\right)}{[X_D^1;\tau_I^1+\sigma_I^1]}$
Goal	$\left[S_D^1; (0, n_d)_I^1\right]$

multitree, whose root is the goal item. Such a tree would contain a $(D - I)$-dimensional translation of the input. Thus, under a boolean semiring, Translator CT can determine whether a translation of the input exists, according to the grammar.

Under an inside-probability semiring, Translator CT can compute the total probability of all D-dimensional multitrees containing the I-dimensional input. All these derivation trees, along with their probabilities, can be efficiently represented as a packed parse forest, rooted at the goal item. Unfortunately, finding the most probable output string still requires summing probabilities over an exponential number of trees. This problem was shown to be NP-hard in the one-dimensional case (Sima'an, 1996). There is no reason to believe that it is any easier in multiple dimensions.

The Viterbi-derivation semiring would be used most frequently in practice. Given a D-PMTG, Translator CT can use this semiring to find the single most probable D-dimensional multitree that covers the I-dimensional input. For example, suppose that

- all the productions in Table 1(a) have probability 1.0, except that Production 8 has probability 0.7 and Production 9 has probability 0.3;

- we employ a uniform cost search strategy, so that the translator makes inferences in order of decreasing probability of the consequent;[8]

- the input is "Pasudu moy."

The inferences that Translator CT would make under these conditions are shown in the proof tree in Figure 6. Each internal node represents an item. The children of each item are its antecedents. The items are numbered to indicate the order of inference. For example, the consequents numbered 3, 5, and 6 precede the one numbered 7, because the former all have probability 1.0, but the probability of the latter is lowered by the probability of its antecedent production rule. The 2nd item is inferred before the 3rd because the label "() MIT" of the 2nd consequent precedes the label "D ()" of the 3rd consequent in the lexicographic order. Note that all the information in the multitree in Figure 3 is also in the proof tree in Figure 6.

One of the productions in Table 1 is absent from Figure 6. By replacing the naive bottom-up search strategy with a more sophisticated one, the translator avoids the expense of an inference involving Production 9. The benefits of alternative search strategies are easier to see when the grammar, the logic, and the semiring are abstracted away and held constant.

The multitree inferred by the translator will have the words of both the input and the output components in its nodes. In practice, we usually want the output as a string tuple, rather than as a multitree. Under the various derivation semirings (Goodman, 1998), Translator CT can store the output precedence arrays ρ_D^{I+1} in each internal node of the tree. The intended ordering of the terminals in each output dimension can be assembled from these templates by a linear-time **linearization** post-process that traverses the finished multitree in postorder.

To the best of our knowledge, Translator CT is the first to be compatible with all of the semirings listed in Section 1.3, among others. It is also unique in being able to accommodate multiple input components and multiple output components. When a source document is available in multiple languages, a translator can benefit from the disambiguating information in each. Translator CT can take advantage of such information without making the strong independence assumptions of Och and Ney (2001). When output is desired in multiple languages, Translator CT offers all the putative benefits of the interlingual approach to MT, including greater efficiency and greater consistency across output components. Indeed, the language of multitrees can be viewed as an interlingua.

5. Synchronization

We have explored inference of I-dimensional multitrees under a D-dimensional grammar, where $D \geq I$. Now we generalize along the other axis of Figure 2(a). It is often useful to infer I-dimensional multitrees without the benefit of an I-dimensional grammar. One application is inducing a parser in one language from a parser in another (Lü, Zhao, and Yang, 2002). The application that is most relevant to this article is bootstrapping an I-dimensional grammar. In theory, it is possible to estimate a PMTG from multitext in an unsupervised manner, starting with a random or uniform distribution over production rules. However, the quality of the parameter estimates is greatly affected by how they are initialized.

A more reliable way to estimate PMTG production probabilities is from a corpus of multitrees — a **multitreebank**.[9] We are not aware of any multitreebanks at this time. The most straightforward way to create one is to parse some multitext using a synchronous parser, such as Parser C. However, if the goal is to bootstrap an I-PMTG, then there is no I-PMTG that can evaluate the G terms in the parser's logic. Our solution is to orchestrate lower-dimensional knowledge sources to evaluate the G terms. Then, we can use the same parsing logic to synchronize multitext into a multitreebank.

To illustrate, we describe a relatively simple synchronizer, under the Viterbi-derivation semiring. Under this semiring, a synchronizer computes the single most probable multitree for a given multitext. If we have no PMTG, then we can use other criteria to evaluate multitrees. These other criteria can be based on partial substructures.

[8] Ties are broken according to the lexicographic order of the consequent labels.
[9] In contrast, a parallel treebank (Han, Han, and Ko, 2001) might contain no information about translational equivalence.

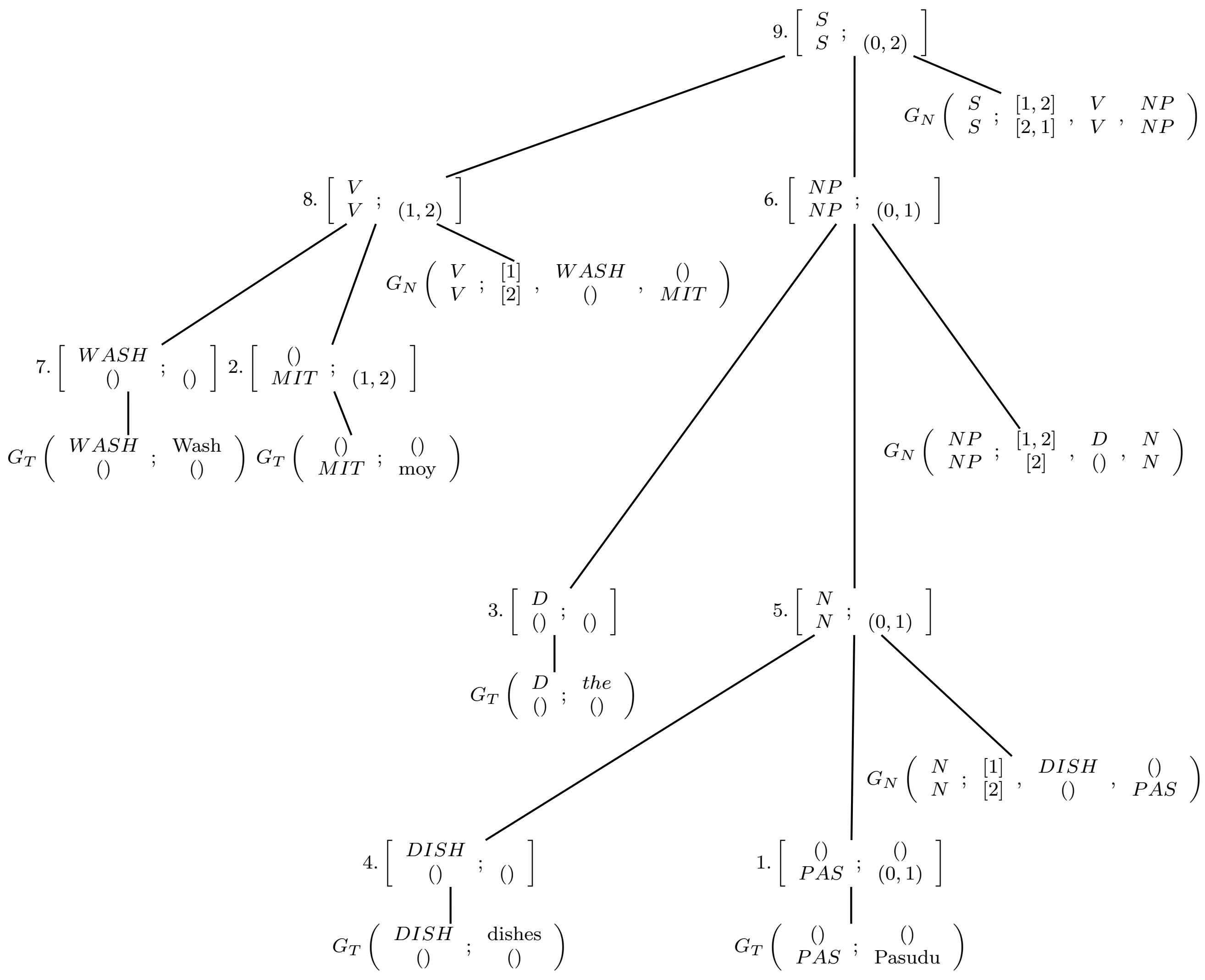


Figure 6
Proof tree for Translator CT's inference of the multitree in Figure 3, using the MTG in Table 1(a) with a Viterbi-derivation semiring, on input "Pasudu moy." The child nodes of each item contain its antecedents. The items are numbered to indicate the order of their inference.

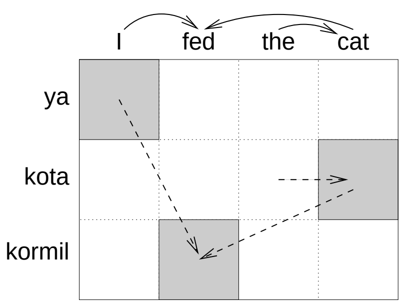


Figure 7
Synchronization. Only one synchronous dependency structure (dashed arcs) is compatible with the monolingual structure (solid arcs) and word alignment (shaded squares).

For example, given a tokenized set of tuples of parallel sentences, it is always possible to estimate a word-to-word translation model $\Pr(u_1^D | u_{D+1}^I)$ (Och and Ney, 2003). [10] Such a probability distribution ranges over parts of the nodes of multitrees. Even if we have no basis for choosing among different syntactic structures, we can prefer multitrees whose individual nodes have higher probability.

We shall consider the common synchronization scenario where a lexicalized monolingual grammar is available for at least one component. For example, many multitexts have at least one component in one of the languages for which treebanks have been built[11]. Given a treebank in the language of one of the input components, we can induce a lexicalized PCFG. Alternatively, if a non-probabilistic parser is available for one of the input components, then we can first parse that component, and then proceed as we would from a treebank. Regardless of how we obtain it, a monolingual lexicalized grammar can guide our search for the multitree with the most probable syntactic structure in the resource-rich component. More generally,[12] we might have a lexicalized PMTG in D dimensions, from which we want to synchronize I-dimensional multitrees, $I > D$. Without loss of generality, we shall let the PMTG range over the first D components. We shall then refer to the D structured components and the $I - D$ unstructured components.

The syntactic structures on different dimensions of a multitree are isomorphic up to reordering of sibling nodes and deletion of subtrees. So, given a fixed correspondence between the words in the nodes of a lexicalized multitree, choosing the optimal structure for one component is tantamount to choosing the optimal synchronous structure for all components.[13] Therefore, a word-to-word translation model and a lexicalized monolingual grammar are sufficient to drive a principled synchronizer. For example, in Figure 7, a monolingual grammar has allowed only one dependency structure on the English side, and a word-to-word translation model has allowed only one word alignment. Ignoring the labels, only one dependency structure is compatible with these constraints.

In order to combine structural and translational constraints in this manner, it is convenient to suppose that we are approximating a *bilexical* PMTG. In a Lexicalized MTG (LMTG) of the bilexical variety (L$_2$MTG) the nonterminal symbols are structured: Every nonterminal has the form $L[t]$ for some terminal $t \in T$ and some label $L \in \Lambda$. Λ is a set of "delexicalized" nonterminal labels. Intuitively, Λ corresponds to the nonterminal set of an ordinary CFG. The terminal t is the **lexical head** of its constituent, or just the **head**. One nonterminal in each component on the RHS of an L$_2$MTG production serves as the **heir** of the nonterminal in the corresponding component on the LHS. The heir inherits the lexical head of its parent nonterminal. An L$_2$MTG has a special start nonterminal $\$$, lexicalized by a special start terminal. The start terminals are deterministically removed from the yield just before the grammar's generative process terminates.

[10] Although most of the literature discusses word translation models between only two languages, it is possible to use one language as a pivot to combine several 2D models into a higher-dimensional model (Mann and Yarowsky, 2001).
[11] At the time of writing, we are aware of treebanks for English, Spanish, French, German, Chinese, Czech, Arabic, and Korean.
[12] Recall that a PCFG is a one-dimensional PMTG.
[13] Except where the unstructured components have constituents that correspond to nothing in the structured components.

Bootstrapping a PMTG from a lower-dimensional PMTG and a word-to-word translation model is similar in spirit to the way that regular grammars can help to induce CFGs (Lari and Young, 1990), and the way that simple translation models can help to bootstrap more sophisticated ones (Brown et al., 1993). Assuming a bilexical PMTG, or a functionally equivalent approximation thereof, we can search for a multitree that simultaneously has a high-probability syntactic structure and a high-probability correspondence among words in its nodes. Such an inference process is, by definition, a generalized parser. It can be based on any synchronous parsing logic, including Logic C. If we have no I-PMTG, then we can redefine the G terms in a way that does not rely on it.

We begin by redefining G_T. For the structured components, we retain the grammar-based definition. I.e., $G_T(X_d[h_d]; h_d) = \Pr(h_d | X_d[h_d])$, where the latter probability can be looked up in a D-PMTG. For the unstructured components, there are no useful nonterminal labels. Therefore, we assume that the unstructured components use only one (dummy) nonterminal label λ, so that $G_T(X_d[h_d]; h_d) = 1$ if $X = \lambda$ and 0 otherwise.

Our treatment of nonterminal productions begins by applying the chain rule[14]

$$\begin{aligned}
G_N(X_I^1[h_I^1]; \rho_I^1, Y_I^1[g_I^1], Z_I^1[h_I^1]) &= \Pr(\rho_I^1, g_I^1, Y_I^1, Z_I^1 | X_I^1, h_I^1) && (19)\\
&= \Pr(\rho_D^1, g_D^1, Y_D^1, Z_D^1 | X_I^1, h_I^1) && (20)\\
&\quad \times \Pr(Y_I^{D+1}, Z_I^{D+1} | \rho_D^1, g_D^1, Y_D^1, Z_D^1, X_I^1, h_I^1) && (21)\\
&\quad \times \Pr(g_I^{D+1} | \rho_D^1, g_D^1, Y_I^1, Z_I^1, X_I^1, h_I^1) && (22)\\
&\quad \times \Pr(\rho_I^{D+1} | \rho_D^1, g_I^1, Y_I^1, Z_I^1, X_I^1, h_I^1) && (23)
\end{aligned}$$

and continues by making independence assumptions. The first assumption is that the structured components of the production's RHS are conditionally independent of the unstructured components of its LHS:

$$\Pr(\rho_D^1, g_D^1, Y_D^1, Z_D^1 | X_I^1, h_I^1) = \Pr(\rho_D^1, g_D^1, Y_D^1, Z_D^1 | X_D^1, h_D^1) \tag{24}$$

The above probability can be looked up in the D-PMTG. Second, since we have no useful nonterminals in the unstructured components, we let

$$\Pr(Y_I^{D+1}, Z_I^{D+1} | \rho_D^1, g_D^1, Y_D^1, Z_D^1, X_I^1, h_I^1) = \begin{cases} 1 \text{ if } Y_I^{D+1} = Z_I^{D+1} = \lambda_I^{D+1} \\ 0 \text{ otherwise} \end{cases}. \tag{25}$$

Third, we assume that the word-to-word translation probabilities are independent of anything else:

$$\Pr(g_I^{D+1} | \rho_D^1, g_D^1, Y_I^1, Z_I^1, X_I^1, h_I^1) = \Pr(g_I^{D+1} | g_D^1) \tag{26}$$

In a typical synchronization scenario, these probabilities would be obtained from a word-to-word translation model, which would be estimated under such an independence assumption.[15] Finally, we assume that the output precedence arrays are independent of each other and uniformly distributed, up to some maximum cardinality m. Let $\upsilon(m)$ be the number of unique precedence arrays of cardinality m or less. Then

$$\Pr(\rho_I^{D+1}, | \rho_D^1, g_I^1, Y_I^1, Z_I^1, X_I^1, h_I^1) = \Pr(\rho_I^{D+1}) = \prod_{d=D+1}^{I} \frac{1}{\upsilon(m)} = \frac{1}{\upsilon(m)^{I-D}}. \tag{27}$$

Under Assumptions 24–27,

$$G_N(X_I^1[h_I^1]; \rho_I^1, Y_I^1[g_I^1], Z_I^1[h_I^1]) = \frac{\Pr(\rho_D^1, g_D^1, Y_D^1, Z_D^1 | X_D^1, h_D^1) \cdot \Pr(g_I^{D+1} | g_D^1)}{\upsilon(m)^{I-D}} \tag{28}$$

[14]The procedure is analogous when the heir is the first nonterminal link on the RHS, rather than the second.

[15]As explained in Section 3, a word-to-word translation model can be viewed as a degenerate MTG. If the D-PMTG used for synchronization is in GCNF, then the word-to-word translation model must also be converted to GCNF for consistency.

if $Y_I^{D+1} = Z_I^{D+1} = \lambda_I^{D+1}$ and 0 otherwise.

In the most common case that the multitext is just a bitext, and we have a structured language model for just one of its components, the above equation boils down to

$$G_N(X_2^1[h_2^1]; \rho_2^1, Y_2^1[g_2^1], Z_2^1[h_2^1]) = \frac{\Pr(\rho_1, g_1, Y_1, Z_1 | X_1, h_1) \cdot \Pr(g_2 | g_1)}{\upsilon(m)} \tag{29}$$

if $Y_2 = Z_2 = \lambda$ and 0 otherwise. We can use these estimates in the inference rules of Logic C, under any probabilistic semiring.

More sophisticated synchronization methods are certainly possible. For example, we could project a part-of-speech tagger (Yarowsky and Ngai, 2001) to improve our estimates in Equation 25. Yet, despite their relative simplicity, the above methods for estimating production rule probabilities use all of the available information in a consistent manner, without double-counting. This kind of synchronizer stands in contrast to more ad-hoc approaches (Matsumoto, Utsuro, and Ishimoto, 1993; Meyers, Yangarber, and Grishman, 1996; Wu and Wong, 1998; Hwa et al., 2002). Some of these previous works fix the word alignments first, and then infer compatible parse structures. Others do the opposite. Information about syntactic structure can be inferred more accurately given information about translational equivalence, and vice versa. Commitment to either kind of information without consideration of the other increases the potential for compounded errors.

6. Multitree-based Statistical Machine Translation

Various cross-lingual applications stand to benefit from the generalizations of parsing described in this article. For example, synchronization is useful for compiling examples for example-based MT (Matsumoto, Utsuro, and Ishimoto and for extracting transfer rules for transfer-based MT (Meyers, Yangarber, and Grishman, 1996). Synchronous parsing is useful for refining the granularity of translation memories for machine-assisted translation. The strong connection between synchronous parsing and translation is also exploited in the Data-Oriented Translation paradigm (Hearne and Way, 2003). Statistical MT is the application that stands to benefit the most.

Figure 8 shows a rudimentary architecture for an SMT system that revolves around multitrees. The generalizations of parsing that play a role in this system are specified by their logic, semiring, and grammar. Any search strategy is acceptable for any of them, at least in theory.[16] The recipe follows:

T1. Induce a word-to-word translation model.

T2. Induce a PCFG from the relative frequencies of productions in the monolingual treebank.

T3. Synchronize some multitext, e.g. using the approximations in Section 5.

T4. Induce an initial PMTG from the relative frequencies of productions in the multitreebank.

T5. Re-estimate the PMTG parameters using a synchronous parser under the expectation semiring.

A1. Use the PMTG to infer the most probable multitree covering the input multitext.

A2. Linearize the output dimensions of the multitree.

Steps T2, T4 and A2 are trivial. Steps T1, T3, T5, and A1 are the generalizations of parsing presented in this article.

Figure 8 shows that generalizations of parsing can do most of the work in a syntax-aware SMT system. An advantage of building an MT system in this manner is that improvements invented for one of these algorithms can often be applied to all of them. For example, Melamed (2003) showed how to reduce the computational complexity of a synchronous parser by a factor of n^D, just by changing the logic. The same

[16] In practice, we must manage computational complexity.

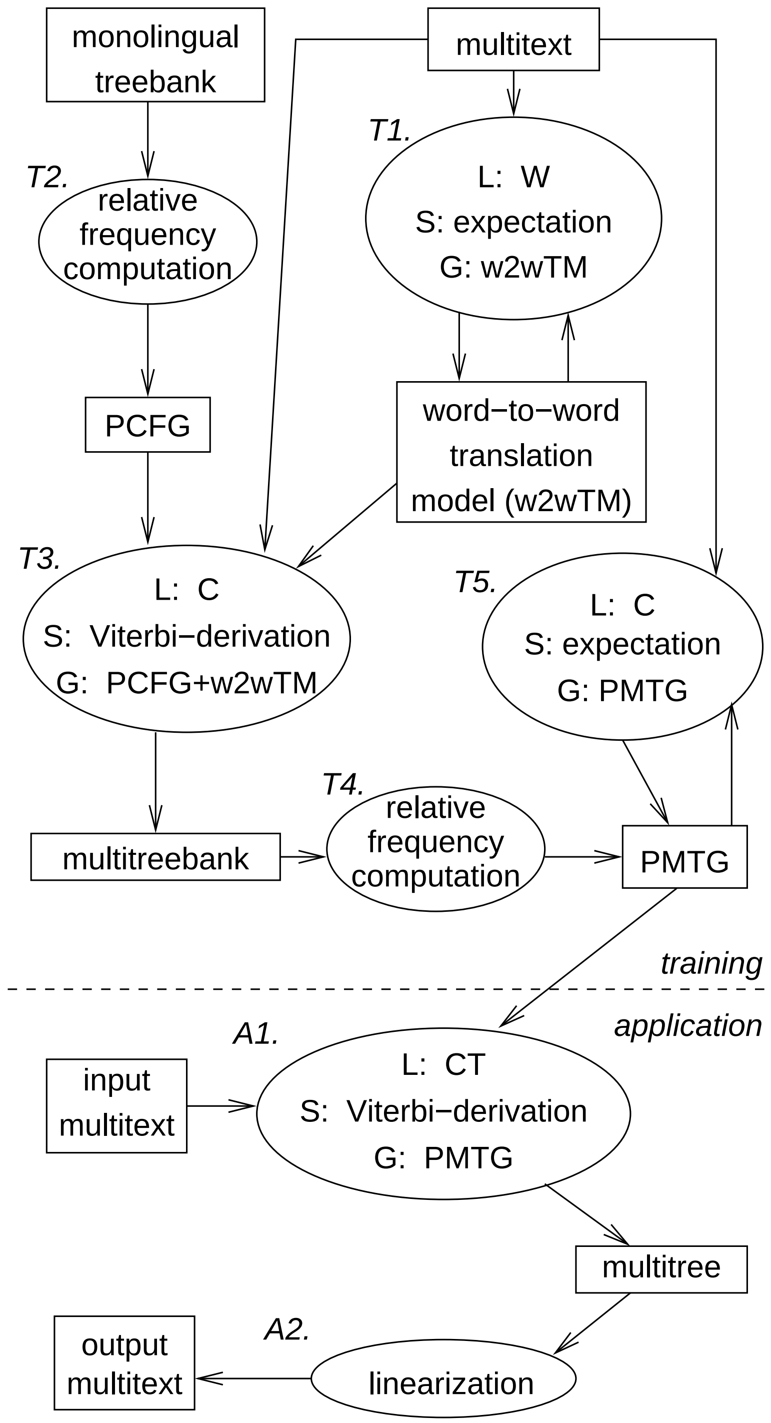


Figure 8
Data-flow diagram for a rudimentary SMT system based on generalizations of parsing. L = logic; S = semiring; G = grammar.

optimization can be applied to any bilexical synchronizer or synchronous parser based on Logic C and to any bilexical translator based on Logic CT. With proper software design, such optimizations need never be implemented more than once.

7. Future Work

The viability of statistical machine translation by parsing depends on research on each of the four components of generalized parsers:

- Grammars. There are many aspects of translational equivalence that MTG can model only clumsily or not at all. Some of these aspects can be modeled by extending MTG in ways that do not increase the asymptotic complexity of the associated inference algorithms. Important examples include non-compositional compounds (Melamed, 1997) and inflectional morphology.
- Logics. All of the logics in this article are based on the "CKY" logic introduced in the algorithms of Kasami (1965) and Younger (1967). Other parsing logics, such as the one in Earley's parser (Earley, 1970), can be substituted throughout. In addition to synchronous generalizations of existing parsing logics, we envision specializations and variations designed specifically for translation.
- Semirings. Maximum likelihood is not the ideal objective function for MT. In practice, maximum likelihood estimators are usually adjusted by various smoothing techniques. A more radical departure from maximum likelihood, based on large-margin methods, has recently been suggested by Collins and Duffy (2001). We conjecture that the best SMT systems will combine these better objective functions with the expressive power of multitrees.
- Search strategies. Analysis by inspection (McAllester, 2002; Melamed, 2003) shows that all of the parsers presented in this article have polynomial complexity, but the exponents of these polynomials are quite high. Clever search strategies are necessary to combat the computational complexity of generalized parsing. However, analogous algorithms for today's best SMT systems have exponential complexity. Though asymptotic complexity is not necessarily indicative of actual performance, we are optimistic that algorithms with polynomial complexity will be easier to optimize or approximate than algorithms with exponential complexity.

In all cases, we will be able to take advantage of past and future research on ordinary grammars and parsers. Therefore, we will be able to concentrate our talents on better models, without worrying about MT-specific "decoders." This reallocation of our collective effort will highlight one of the key advantages of MT by parsing over MT by finite-state transduction: Finite-state translation models run counter to our intuitions about how expressions in different languages are related. MT research based on such models may be a necessary stepping stone but, in the long term, the price of implausible models is reduced insight. Though incremental improvements are possible by hill-climbing on objective criteria, major advances can come only from deep intuitions about the relationship of the models to the phenomena being modeled.

Despite the intuitive appeal and theoretical advantages of syntax-aware SMT systems, it will take some time for them to outperform finite-state ones. After all, the latter class of systems has had a head-start of over a decade.

8. Conclusion

This article has decomposed standard parsing algorithms into a grammar, a logic, a semiring, and a search strategy. It then presented the generalizations of ordinary parsing that emerge when the dimensionality of the grammar and/or the input are allowed to rise. Along the way, it has elucidated the relationships between parsing and other classes of algorithms that previously seemed unrelated or less directly related. It turns out that, given tokenized multitext and a monolingual treebank, a rudimentary syntax-aware statistical machine translation system can be built and applied using only generalized parsers and some trivial glue. This approach to building MT systems encourages MT research to be less specialized and more transparently related to the rest of computational linguistics. We expect generalized parsers to be useful for other problems with a similar structure, such as sentence compression (Knight and Marcu, 2000) and syntax-aware generation (Langkilde, 2000).

Acknowledgments

Many thanks to Joseph Turian, Ben Wellington, and the anonymous reviewers for valuable feedback. This research was supported by an NSF CAREER Award, by the DARPA TIDES program, and by an equipment gift from Sun Microsystems.

References

[Aho and Ullman1969] 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, Bangalore, and Douglas2000] 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.

[Brown et al.1993] Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and Rober L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):45–60.

[Collins and Duffy2001] Collins, Michael and Nigel Duffy. 2001. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, July.

[Earley1970] Earley, Jay. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

[Eisner2002] Eisner, Jason. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, July.

[Gildea2003] Gildea, Daniel. 2003. Loosely tree-based alignment for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 80–87, Sapporo, Japan, July.

[Goodman1998] Goodman, Joshua. 1998. *Parsing Inside-Out*. Ph.D. thesis, Harvard University.

[Han, Han, and Ko2001] Han, Chung-Hye, Ra-Rae Han, and Eon-Suk Ko. 2001. Bracketing guidelines for the penn korean treebank. Technical Report IRCS-01-010, IRCS, University of Pennsylvania.

[Hearne and Way2003] Hearne, Mary and Andy Way. 2003. Seeing the wood for the trees: Data-oriented translation. In *Proceedings of Machine Translation Summit IX*.

[Hwa et al.2002] Hwa, Rebecca, Philip Resnik, Amy Weinberg, and Okan Kolak. 2002. Evaluating translational correspondence using annotation projection. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, July.

[Johnson1985] Johnson, Mark. 1985. Parsing with discontinuous constituents. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 127–132, Chicago, Illinois, USA, July.

[Jurafsky and Martin2000] Jurafsky, Daniel and James H. Martin. 2000. *Speech and Language Processing*. Prentice-Hall.

[Kasami1965] Kasami, Tadao. 1965. An efficient recognition and syntax algorithm for context-free languages. Technical Report AF-CRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA.

[Klein and Manning2003] Klein, Dan and Christopher D. Manning. 2003. A* parsing : Fast exact viterbi parse selection. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, Edmonton, Canada.

[Knight and Köhn2003] Knight, Kevin and Philipp Köhn. 2003. What's new in statistical machine translation. In *Tutorial presented at Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*. Available at `http://www.isi.edu/~koehn/publications/tutorial2003 .pdf`.

[Knight and Marcu2000] Knight, Kevin and Daniel Marcu. 2000. Statistics-based summarization — step one: Sentence compression. In *Proceedings* *of the American Association for Artificial Intelligence (AAAI)*.

[Kumar and Byrne2003] Kumar, Shankar and William Byrne. 2003. A weighted finite state transducer implementation of the alignment template model for statistical machine translation. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 63–70, Edmonton, Canada.

[Langkilde2000] Langkilde, Irene. 2000. Forest-based statistical sentence generation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, Seattle, May.

[Lari and Young1990] Lari, K. and S. J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language Processing*, 4:35–56.

[Lü, Zhao, and Yang2002] Lü, Yajuan, Tiejun Zhao, and Muyun Yang. 2002. Learning Chinese bracketing knowledge based on a bilingual language model. In *Proceedings of the International Conference on Computational Linguistics 2002*, pages 591–597, Taipei, Taiwan, August.

[Mann and Yarowsky2001] Mann, Gideon S. and David Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *Proceedings of the Applied Natural Language Processing and the North American Chapter of the Association for Computational Linguistics (ANLP-NAACL)*, pages 151–158, Pittsburgh, June.

[Matsumoto, Utsuro, and Ishimoto1993] Matsumoto, Yuji, Takehito Utsuro, and Hiroyuki Ishimoto. 1993. Structural matching of parallel texts. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, Columbus, Ohio, June.

[McAllester2002] McAllester, David. 2002. On the complexity analysis of static analysis. *Journal of the ACM*, 49(4).

[Melamed1997] Melamed, I. Dan. 1997. Automatic discovery of non-compositional compounds. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, Providence, RI.

[Melamed2003] Melamed, I. Dan. 2003. Multitext grammars and synchronous parsers. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, Edmonton, Canada.

[Melamed, Satta, and Wellington2004] Melamed, I. Dan, Giorgio Satta, and Benjamin Wellington. 2004. Generalized multitext grammars. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain, July.

[Meyers, Yangarber, and Grishman1996] Meyers, Adam, Roman Yangarber, and Ralph Grishman. 1996. Alignment of shared forests for bilingual corpora. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, Denmark.

[Och and Ney2001] Och, Franz Josef and Hermann Ney. 2001. Statistical multi-source translation. In *Proceedings of Machine Translation Summit VIII*, pages 253–258, Santiago de Compostela, Spain, September.

[Och and Ney2002] Och, Franz Josef and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Philadelphia, July.

[Och and Ney2003] Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, March.

[Rambow and Satta1999] Rambow, Owen and Giorgio Satta. 1999. Independent parallelism in finite copying parallel rewriting systems. *Theoretical Computer Science*, 223:87–120.

[Shieber, Schabes, and Pereira1995] Shieber, Stuart M., Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–36.

[Sima'an1996] Sima'an, Khalil. 1996. Computational complexity of probabilistic disambiguation by means of tree-grammars. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, Denmark.

[Wu1996] Wu, Dekai. 1996. A polynomial-time algorithm for statistical machine translation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz, CA, June.

[Wu1997] Wu, Dekai. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404, September.

[Wu and Wong1998] Wu, Dekai and Hongsing Wong. 1998. Machine translation with a stochastic grammatical channel. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, Montreal, Canada, July.

[Yamada and Knight2002] Yamada, Kenji and Kevin Knight. 2002. A decoder for syntax-based statistical mt. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, July.

[Yarowsky and Ngai2001] Yarowsky, David and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of the Applied Natural Language Processing and the North American Chapter of the Association for Computational Linguistics (ANLP-NAACL)*, Pittsburgh, June.

[Younger1967] Younger, Daniel H. 1967. Recognition and parsing of context free languages in time n^3. *Information and Control*, 10(2):189–208.