Efficient Outside Computation

Daniel Gildea
University of Rochester
Computer Science Department
gildea@cs.rochester.edu

Weighted deduction systems provide a framework for describing parsing algorithms that can be used with a variety of operations for combining the values of partial derivations. For some operations, inside values can be computed efficiently, but outside values cannot. We view outside values as functions from inside values to the total value of all derivations, and we analyze outside computation in terms of function composition. This viewpoint helps explain why efficient outside computation is possible in many settings, despite the lack of a general outside algorithm for semiring operations.

1. Introduction

In weighted deduction systems such as those used for parsing with context-free grammars, the inside–outside algorithm provides an efficient way of finding the total weight of all derivations passing through a specific item. Weighted deduction systems can be used with different semirings, or even more generally, with other classes of functions for computing the values of items bottom–up in the inside pass. In some cases, efficient inside computation is possible, but efficient outside computation is not. How can these cases be characterized?

We give a very general characterization of the conditions for efficient outside computation in terms of function composition, as well as three more specific examples of sufficient conditions. The first of these conditions, commutative semirings, is discussed by Goodman (1999), while we believe the other two, extremal semirings and the sum of linear functions, to be novel formulations. We discuss general superior functions as a case where efficient outside computation is not possible. We conclude that, despite the emphasis in the literature on describing weighted deduction in terms of semirings, semirings are not the best abstraction for describing the requirements of the general inside–outside algorithm.

2. Weighted Deduction

A **weighted deduction system** (Nederhof 2003) has **rules** of the form $\frac{A_1, \ldots, A_n}{C}$ where $A_1, \ldots, A_n$ are the **items** of the system that form the **antecedents** of the rule, and C is an item that forms the **consequent** of the rule. One item is designated as the **goal** of

Submission received: 1 May 2020; revised version received: 7 July 2020; accepted for publication: 21 September 2020.

https://doi.org/10.1162/COLI_a_00386

the system. Associated with each rule R is a function F_R which takes the **weights** of the antecedent items, and calculates a new weight. A **derivation** is a tree of rules where the antecedents of each rule are the consequents of its children. The leaves of this tree are rules having zero antecedents, also referred to as **axioms**. The weight of a derivation is computed by recursively evaluating the functions F_R; that is, for a derivation D formed by applying rule R to derivations $D_1, \ldots, D_n$:

$$\text{weight}(D) = F_R(\text{weight}(D_1), \ldots, \text{weight}(D_n)) \tag{1}$$

The fundamental problem associated with weighted deduction systems is to find the total weight of all derivations of the goal item. This total weight is computed with a generalized sum operation that will be denoted by $\oplus$.

Weighted deduction systems provide a general framework for expressing and reasoning about dynamic programming algorithms, and in particular about parsing algorithms (Shieber, Schabes, and Pereira 1995; Sikkel 1997; Nederhof 2003). The deduction rule for the basic combination step of CYK parsing of a context-free grammar (CFG) is shown in Figure 1(a). The goal item for CFG parsing with start symbol S and sentence length n is $[S; 0; n]$, where i, j, and k range over positions in a string. In order to simplify our definition of weighted deduction systems, we include the CFG rule $S \rightarrow A\,B$ as an antecedent of the rule, although it is sometimes also represented as a "side condition" for the rule, as in Nederhof (2003), in which case the weight w_1 of the rule can be incorporated into the function F_R. Weighted deduction systems can be used to express other parsing algorithms, including Earley parsing and dependency parsing (Eisner and Satta 1999). Beyond CFG, weighted deduction systems are used for parsing for tree adjoining grammars (Alonso et al. 1999), combinatory categorical grammars (Kuhlmann and Satta 2014), and general linear context-free rewriting systems (Burden and Ljunglöf 2005), as well as for machine translation (Melamed, Satta, and Wellington 2004; Lopez 2009). In all of these applications, a set of general deduction rules is instantiated into a **hypergraph** for a specific input string. For example, given a sentence of length n, the general rule is shown in Figure 1(a). The goal item for CFG parsing with start symbol S and sentence length n $[S; 0; n]$ is instantiated into a specific rule for each combination of $i, j, k \in \{0, \ldots, n\}$. Each instantiated item is a vertex in the hypergraph, and each instantiated rule is a hyperedge from the antecedent vertices to the consequent vertex. The resulting hypergraphs are also known as parse forests. In this article, we will deal exclusively with deduction systems that are already instantiated into hypergraphs. We will refer to hyperedges simply as edges. We use E to refer to the set of edges (instantiated rules), and $|E|$ to refer to the number of edges. For CYK parsing of a string of length n with a set of CFG productions P, $|E| \in O(|P|n^3)$. However, our

(a)

$$\begin{array}{ll} w_1\text{:} & A_1 \\ w_2\text{:} & A_2 \\ w_3\text{:} & A_3 \\ \hline F_R(w_1, w_2, w_1)\text{:} & C \end{array}$$

(b)

$$\begin{array}{ll} w_1\text{:} & [S \rightarrow A\,B] \\ w_2\text{:} & [A, i, j] \\ w_3\text{:} & [B, j, k] \\ \hline F_R(w_1, w_2, w_3)\text{:} & [S, i, k] \end{array}$$

Figure 1
A general weighted deduction rule, and a rule of CFG parsing in weighted deduction notation. The goal item for CFG parsing with start symbol S and sentence length n is $[S, 0, n]$.

discussion will apply equally to the various other applications of weighted deduction systems just mentioned. To simplify the presentation, we will assume at first that our deduction system does not have cycles, that is, an item cannot appear as the consequent of any derivation in which it also appears as an antecedent. For parsing CFGs, this is true whenever the grammar is in Chomsky Normal Form. We return to discuss systems with cycles in Section 4.

Efficient computation on weighted deduction systems depends on a general dynamic programming algorithm that computes a table of **inside values** for each item. The inside value of an item B represents the total weight of all derivations of B.

$$V(B) = \bigoplus_{\text{derivations } D \text{ of } B} \text{weight}(D)$$

The general inside algorithm computes the table of inside values efficiently by summing over rules R having B as a consequent, and applying the function F_R to the (previously computed) inside values of the rule's antecedents.

$$V(B) = \bigoplus_{R: \frac{A_1, \ldots, A_n}{B}} F_R(V(A_1), \ldots, V(A_n)) \tag{2}$$

Items are sorted in a topological order to ensure that inside values of an antecedent are ready before the calculation of the consequent.

The basic property that is required to enable dynamic programming is that the generalized sum must distribute over each argument:

$$\forall i, \bigoplus_{x_i} F_R(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) = F_R(x_1, \ldots, x_{i-1}, \bigoplus_{x_i} x_i, x_{i+1}, \ldots, x_n) \tag{3}$$

Weighted deduction can be performed with various choices of F_R and $\oplus$. The most common choices are the max-product or Viterbi algorithm, where weights are nonnegative real numbers, F_R is always multiplication (regardless of the rule R), and $\oplus$ is the maximum operation. The sum-product algorithm, used to derive the total probability of a string, or as a subroutine of the Expectation Maximization (EM) algorithm (Dempster, Laird, and Rubin 1977), is the case where weights are real numbers, F_R is always multiplication, and $\oplus$ is addition. In the case of CYK parsing, using the sum-product algorithm, the inside recurrence of Equation (2) takes the familiar form:

$$V([A, i, k]) = \sum_{B,C,j} P(A \rightarrow B\,C)\, V([B, i, j])\, V([C, j, k])$$

because the set of rules with item $[A, i, k]$ as a consequent can be found by iterating over nonterminals B and C and split points j. Every deduction rule has three antecedents; the inside value of the axiom $A \rightarrow B\,C$ is defined as the grammar's probability $P(A \rightarrow B\,C)$, and the function F_R simply multiplies these three inside values. The sum-product algorithm was the focus of the first presentations of the inside–outside algorithm for parsing by Baker (1979) and Lari and Young (1990). Eisner (2016) relates the sum-product inside–outside algorithm to backpropagation as used in neural networks (Rumelhart, Hinton,

and Williams 1986) by showing that the inside–outside algorithm can be derived with automatic differentiation.

The max-product and sum-product algorithms are both instances of the semiring parsing framework of Goodman (1999). A **semiring** over a set $\mathbb{K}$ consists of two operations $\oplus$ and $\otimes$ such that:

- $\oplus$ is associative and commutative, and has an identity element $\bar{0}$
- $\otimes$ is associative and has an identity element $\bar{1}$
- $\otimes$ distributes over $\oplus$, and
- for all $x \in \mathbb{K}$, $\bar{0} \otimes x = x \otimes \bar{0} = \bar{0}$

The semiring parsing framework uses these operators to combine partial derivations, and is an instance of our general definition of weighted deduction. For all rules R, the function F_R is the semiring product of its arguments:

$$F_R(x_1, \ldots, x_n) = \bigotimes_{i=1}^{n} x_i \tag{4}$$

Applying our general recurrence of Equation (2), the inside value of an item is the semiring sum over rules producing the item of the semiring product of each rule's antecedents:

$$V(B) = \bigoplus_{R: \frac{A_1, \ldots, A_n}{B}} \bigotimes_{i=1}^{n} V(A_i)$$

Goodman (1999) describes a number of other semirings that can be used in this general algorithm. In particular, the **Viterbi derivation semiring**, discussed in more detail in Section 3.2, computes the value of the highest weight derivation along with a record of the derivation itself. The **derivation semiring** collects a set of all valid derivations. The size of this set can be exponential in the number of edges in the hypergraph.

As an alternative to the semiring framework, Knuth (1977) defines **superior functions** to be functions that are monotonically increasing in each argument, and that have the property that the function is greater than or equal to each of its arguments. In Knuth's framework, each F_R can be any superior function, and the generalized sum for weighted deduction is the minimum operation:

$$V(B) = \min_{R: \frac{A_1, \ldots, A_n}{B}} F_R(V(A_1), \ldots, V(A_n))$$

Defining F_R to be the sum of its arguments yields an algorithm that is an instance of both the semiring framework and the superior function framework. This min-sum algorithm is equivalent to max-product (Viterbi) if we transform each value by taking its negative logarithm. However, the superior function formulation also includes functions such as $F(x_1, x_2) = x_1 + \exp(x_2)$ that are not associative, as well as functions with more than two arguments. The sum-product algorithm, on the other hand, is an instance of the semiring framework, but is not an instance of the superior function framework, because the generalized sum is not the minimum operation.

In general, one can allow items to have weights of different types, for example, vectors of various dimensions. Dynamic programming is possible as long as each type has a generalized sum operation, and as long as Equation (3) holds for each rule, with the first sum interpreted as the sum operator for the type of the rule's consequent, and the second sum interpreted as the sum operator for the type of the ith antecedent.

3. Outside Computation

We refer to the total value of all derivations passing through an item X as the item's **total weight** $\gamma(X)$:

$$\gamma(X) = \bigoplus_{D: X \in D} \text{weight}(D) \tag{5}$$

where each derivation D consists of a complete tree of rules having the goal item as a consequent, and $X \in D$ means that item X is the consequent of some rule in D. Note that the total weight is defined over complete derivations, unlike inside values.

A semiring is called commutative if its multiplication operator is commutative: $a \otimes b = b \otimes a$. When using a commutative semiring in the semiring framework, an **outside value** $Z(X)$ for an item X is a value that can be combined with the inside value to obtain the total weight of an item:

$$\gamma(X) = Z(X) \otimes V(X) \tag{6}$$

Outside values with the sum-product semiring are used to compute expected counts of grammar rules in the EM algorithm. Outside values in the max-product semiring can be used to find the highest probability parse that includes a given item. For example, for CYK parsing of a sentence $w_1 \cdots w_n$ with the max-product semiring, the outside value $Z([A,i,j])$ would be the value of the highest scoring parse tree having the grammar's start symbol as the root, and the string $w_1 \cdots w_i A w_{j+1} \cdots w_n$ as leaves. The product $Z([A,i,j])\, V([A,i,j])$ is the score of the best tree generating $w_1 \cdots w_n$ and containing a node A over the substring $w_{i+1} \cdots w_j$.

For commutative semirings, the outside value $Z(X)$ for an item X can be computed by iterating over rules R that take X as an antecedent, and multiplying the outside value of R's consequent with the inside values of R's other antecedents:

$$Z(X) = \bigoplus_{\substack{R, i: \\ R = \frac{A_1, \ldots, A_n}{B} \\ A_i = X}} Z(B) \otimes \bigotimes_{\substack{j=1, \\ j \neq i}}^{n} V(A_j) \tag{7}$$

For CYK parsing with the sum-product algorithm, the recurrence above corresponds to the standard recurrence for outside probabilities:

$$\begin{aligned} Z([B,i,j]) = &\sum_{A,C,k} Z([A,i,k])\, P(A \rightarrow B\, C)\, V([C,j,k]) + \\ &\sum_{A,C,k} Z([A,k,j])\, P(A \rightarrow C\, B)\, V([C,k,i]) \end{aligned}$$

The first sum includes rules of the form shown in Figure 1(a). The goal item for CFG parsing with start symbol S and sentence length n is $[S;0;n]$ where $[B,i,j]$ is the

second of the three antecedents, and the second sum includes rules where it is the third antecedent.

Outside values can be efficiently computed with a top–down or outside pass through the deduction system after first performing a bottom–up pass to compute the inside values for each item.

We depend on the fact that the $\otimes$ operation is commutative, because we re-order the product $V(A_1) \otimes \cdots \otimes V(A_n)$ by removing $V(A_i)$, in order to later multiply it in from the right in Equation (6).

For non-commutative semirings, the situation is more complex, because one must combine values in the correct order. Goodman (1998, Section 2-C) defines a new semiring, defined from an arbitrary inside semiring, for outside computation. The values of this new outside semiring are sets of pairs of values from the inside semiring. Although this approach shows that there is a semiring that can be used for outside computation, Goodman does not give a general, efficient algorithm for computing outside values. The values in the outside semiring may grow exponentially large (because they are sets of pairs), making the general inside–outside algorithm exponential even when operations on the inside semiring are efficient.

We wish to give a general set of conditions under which efficient outside computation is possible, and to specify the general algorithm. Let us first state the problem by giving a precise definition of efficient outside computation. We will use $|\gamma(X)|$ to indicate the size of the representation (in memory) of $\gamma(X)$.

Definition 1
Given a weighted deduction system, let g be a function such that, as the size $|E|$ of the system's instantiated hypergraphs grows, $\max_X |\gamma(X)| \in O(g(|E|))$. **Efficient outside computation** refers to any algorithm that computes the total weight $\gamma(X)$ of all items X in time $O(|E|\, g(|E|))$.

We include the term $g(|E|)$ in our definition in order to cover situations such as the derivation semiring, where the size required for the goal item G, $|\gamma(G)|$, is exponential in $|E|$, and $|\gamma(G)|$ provides an upper bound on $|\gamma(X)|$ for all items X. However, in most cases, and in all the examples discussed in this article, $g(|E|)$ can be treated as a constant. In this case, efficient outside computation is equivalent to time linear in the size of the hypergraph.

Our definition of efficient outside computation does not explicitly require a top–down or outside pass through the deduction system. It is possible in some settings to compute the total weight of an item without an outside pass. For example, in CYK parsing, one can first eliminate all items not consistent with a fixed item denoting a particular pair of nonterminal and span, and one can then compute the total weight of all remaining derivations bottom–up (Pereira and Schabes 1992), as shown in Algorithm 1.

Algorithm 1 Bottom–Up Computation of Total Weight

```
procedure BottomUpTotalWeight
    for Items [A, i, j] do
        Remove all items [A', i', j'] such that i < i' < j < j' or i' < i < j' < j
        Compute inside value of goal V(G) = V([S, 0, n])
        γ([A, i, j]) ← V(G)
        Restore all items previously removed
```

For CYK parsing $|E| \in O(n^3)$ with respect to the sentence length n. The outer loop of Algorithm 1 has $O(n^2)$ iterations, and each inside pass is $O(n^3)$, for a total runtime of $O(n^5)$. Thus, using this method to compute the total weight for *all* items in the system takes time greater than $O(|E|\, g(|E|))$. Computing the total weight of all items is necessary for the EM algorithm, perhaps the most common use case for outside computation. As another use case, one may also wish to precompute the best derivation passing through each item, using the Viterbi semiring, in order to be able to later look up in constant time the best derivation for any desired item. Our definition of efficient outside computation is chosen so as not to predetermine any specific algorithm, but also to rule out less efficient procedures such as repeated bottom–up computation.

In our general framework of weighted deduction, each derivation corresponds to a tree of function evaluations. The outside value of an item X can be thought of as a function $F_{\neg X}$ from the inside value of X to the total weight of X:

$$F_{\neg X}(V(X)) = \gamma(X)$$

We call $F_{\neg X}$ defined above an **outside function**. We use the symbol $\neg$ as a mnemonic for "outside" in the definition above and in the remainder of this article. In the commutative semiring framework, the outside function multiplies its argument with the outside value $Z(X)$ discussed above:

$$F_{\neg X}(x) = Z(X) \otimes x \tag{8}$$

The outside function $F_{\neg X}$ can also be formulated in terms of paths through the deduction system, as shown in Figure 2. We refer to a sequence of deductions $R_1, \ldots, R_n$ such that the consequent of each R_i is an antecedent of R_{i+1} as a **path** p. Let $R_i = \dfrac{A_{i,1}, \ldots, A_{i,n_i}}{C_i}$ with $C_{i-1} = A_{i,j_i}$, that is, j_i specifies which antecedent of rule R_i is satisfied by the consequent of rule R_{i-1}. We define a function f_i for each rule on the path p by fixing the inside values of the other antecedents, and projecting the rule's inside function onto argument j_i:

$$f_{p,i}(x) = F_{R_i}(V(A_{i,1}), \ldots, V(A_{i,j_i-1}), x, V(A_{i,j_i+1}), \ldots, V(A_{i,n_i})) \tag{9}$$

The outside function $F_{\neg X}$ can be expressed as a sum over paths from item X to the goal item G.

$$F_{\neg X}(x) = \bigoplus_{\text{paths } p \text{ from } X \text{ to } G} f_{p,n} \circ \cdots \circ f_{p,1}(x) \tag{10}$$

This can be shown by taking the sum over all derivations through item X, grouping the derivations according to the path from X to the goal G, and applying the general distributive rule for weighted deduction of Equation (3):

$$\gamma(X) = \bigoplus_{D:X \in D} \text{weight}\,(D) \tag{11}$$

$$= \bigoplus_{\text{paths } p \text{ from } X \text{ to } G} \; \bigoplus_{D:p \subseteq D} \text{weight}\,(D) \tag{12}$$

$$= \bigoplus_{\text{paths } p \text{ from } X \text{ to } G} f_{p_n} \circ \cdots \circ f_{p_1}(\, V(X)\,) \tag{13}$$

$$= F_{\neg X}(\, V(X)\,) \tag{14}$$

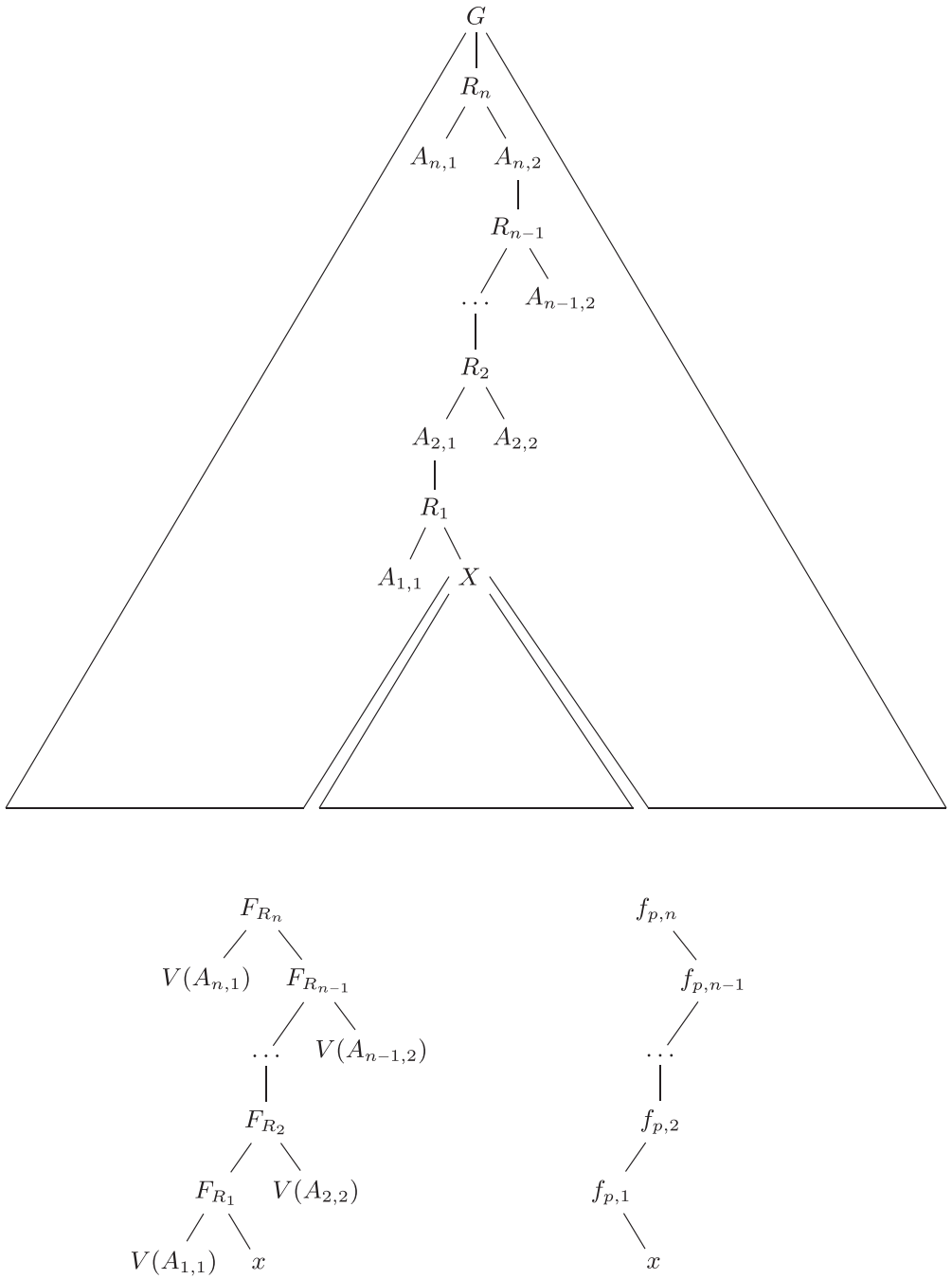

Figure 2
Any tree outside an item X contains a path from X to the goal item G (top). Each rule along the path specifies a function, which can be applied to the inside values of the rule's other antecedents (bottom left). Composing the resulting unary functions along the path results in the outside function $F_{\neg X}$ (bottom right).

Standard algorithms for outside computation compute this sum of function compositions using dynamic programming. The top–down, outside pass of computation consists of creating a representation of $F_{\neg X}$ from the set of the representations of $F_{\neg B}$ for all items B that are consequents of a rule having X as an antecedent:

$$F_{\neg X}(x) = \bigoplus_{\substack{R,i: \\ R=\frac{A_1,\ldots,A_n}{B} \\ A_i=X}} F_{\neg B}(F_R(V(A_1),\ldots,V(A_{i-1}),x,V(A_{i+1}),\ldots,V(A_n))) \tag{15}$$

For commutative semirings, this recurrence is equivalent to Equation 7, as can be seen by substituting in Equation (8) for $F_{\neg B}$ and Equation 4 for F_R. For non-commutative semirings, by induction on the length of the paths, we see that:

$$F_{\neg X}(x) = \bigoplus_{p} a_p \otimes x \otimes b_p \tag{16}$$

where p ranges over all paths from X to the goal item, and a_p and b_p are semiring values determined from the inside values along path p. However, the exponentially large number of terms in the sum may make outside computation difficult.

The formulation of Equation (15) leads to a simple general condition for efficient outside computation.

Theorem 1
Let $\texttt{out}(X)$ be the set of items B such that some rule has X as an antecedent and B as a consequent, and define $g(|E|)$ as in Definition 1. Efficient outside computation is possible if a representation of $F_{\neg X}$ can be computed with $|\texttt{out}(X)|$ operations of time $O(g(|E|))$, given $F_{\neg B}$ for each $B \in \texttt{out}(X)$, and if the representation can be evaluated in time $O(g(|E|))$.

Proof. Procedure OUTSIDE (Algorithm 2) computes $F_{\neg X}$ for all items X using time $\sum_X |\texttt{out}(X)|\, O(g(|E|))$. The sum $\sum_X |\texttt{out}(X)|$ is bounded by summing the number of antecedents for each in E, so $\sum_X |\texttt{out}(X)| \in O(|E|)$, yielding total time $O(|E|\, g(|E|))$ for Algorithm 2. We then compute $\gamma(X) = F_{\neg X}(V(X))$ in time $O(|E|\, g(|E|))$, satisfying the conditions of Definition 1. □

Algorithm 2 General Outside Pass

```
procedure OUTSIDE
    for Items X in reverse topological order do
        Compute F_¬X with Equation (15)
```

We will give examples of settings that do and that do not meet this general criterion for efficient outside computation.

3.1 Commutative Semirings

For any commutative semiring, the representation of the outside function $F_{\neg X}(x)$ consists of the outside value $Z(X)$. If semiring operations take time $O(g(|E|))$, this value can be computed for all items X in time $O(|E|\, g(|E|))$ using Equation (7). The outside function

can be evaluated with a single semiring multiplication using Equation (8). Therefore the conditions of Theorem 1 are met, yielding the following corollary:

Corollary 1
Efficient outside computation is possible for any commutative semiring whose operations can be computed in time $O(g(|E|))$.

In particular, efficient outside computation is possible whenever semiring operations take constant time. The general outside pass of Algorithm 2 takes the following form for commutative semirings.

Algorithm 3 Outside Pass for Commutative Semiring

```
procedure OUTSIDECOMMUTATIVE
    for Items X in reverse topological order do
        Compute Z(X) with Equation (7)
        Set F_{¬X}(x) = Z(X) ⊗ x
```

Commutative semirings include the sum-product semiring used for finding the total probability of all parses of a string, as well as the max-product and max-sum (Viterbi) semirings used for finding the score of the best parse. Other examples include: the K-best semiring used to find the scores for the k best parses (Mohri 2002), the expectation semiring used to compute expected feature values for EM or for training log-linear models (Eisner 2002), the variance semiring used in minimum risk training of log-linear models (Li and Eisner 2009), the entropy semiring used to compute the entropy of the distribution over parses (Hwa 2004; Cortes et al. 2006), the generalized entropy semiring used to compute the relative entropy between two grammars (Cohen, Simmons, and Smith 2011), and the k-best+residual semiring used to find the k best scores and total score simultaneously (Gimpel and Smith 2009). Gimpel and Smith (2009) also define "generalized" semirings for approximate inference that do not meet all the criteria that define a semiring, but that have a commutative $\otimes$ operator and thus admit outside computation with Algorithm 3.

3.2 Extremal Semirings

A semiring is **extremal** if for all a, b, either $a \oplus b = a$ or $a \oplus b = b$ (Vorobev 1963). The max-product semiring is extremal, as is any semiring over real numbers having max as the generalized addition operator. An extremal semiring is always idempotent, meaning that $a \oplus a = a$.

Another example of an extremal semiring is the Viterbi derivation semiring of Goodman (1999). Values in this semiring consist of a pair whose first item is a real number, and whose second item is a record of a partial derivation. This semiring is used to find a maximum scoring derivation, rather than merely computing the maximum score as a real number. The record of the partial derivation can be implemented with back pointers; this semiring is a mathematical formalization of the standard use of backpointers in dynamic programming algorithms. The semiring operation $a \oplus b$ returns whichever of a and b has the highest value as the first element (score) of the pair. The operation $a \otimes b$ multiplies the scores of a and b and concatenates the derivations

into a new derivation. This semiring is non-commutative, because the concatenation in the $\otimes$ operator is non-commutative. The Viterbi derivation semiring is extremal.[1]

For extremal semirings, it is sufficient to retain the outside value of a single outside derivation. We will now prove this fact and derive a general algorithm for extremal semirings. The **natural order** of a semiring is defined by:

$$\begin{aligned} a \leq b \quad & \text{if} \quad a \oplus b = b \\ b \leq a \quad & \text{if} \quad a \oplus b = a \end{aligned}$$

The natural order of an extremal semiring is a total order, because one of the two cases above applies for any pair of a and b.

An extremal semiring is monotonic with respect to its natural order, meaning that:

$$\begin{aligned} a \leq b \quad & \Longrightarrow \quad a \otimes c \leq b \otimes c \\ a \leq b \quad & \Longrightarrow \quad c \otimes a \leq c \otimes b \end{aligned}$$

For a short proof, see Lemma 2 of Mohri (2002). Monotonicity implies that:

$$a \leq b \quad \Longrightarrow \quad (a \otimes c) \oplus (b \otimes c) = (b \otimes c) \tag{17}$$

$$a \leq b \quad \Longrightarrow \quad (c \otimes a) \oplus (c \otimes b) = (c \otimes b) \tag{18}$$

We now show that the outside function of an item X can be represented by one left and one right multiplication:

$$F_{\neg X}(x) = a \otimes x \otimes b \tag{19}$$

To see this, let B range over consequents of rules with X as an antecedent, and assume as an inductive hypothesis that B's outside function can be represented as one left and one right multiplication:

$$F_{\neg B}(x) = a_B \otimes x \otimes b_B$$

Applying the composition rule of Equation (15) yields:

$$F_{\neg X}(x) = \bigoplus_{\substack{R,i: \\ R=\frac{A_1 \ldots, A_n}{B} \\ A_i = X}} a_B \otimes \left(\bigotimes_{j=1}^{i-1} V(A_j) \right) \otimes x \otimes \left(\bigotimes_{j=i+1}^{n} V(A_j) \right) \otimes b_B \tag{20}$$

which can be summarized by

$$F_{\neg X}(x) = \bigoplus_i a_i \otimes x \otimes b_i \tag{21}$$

for the appropriate choice of a_i and b_i.

1 In order to break ties between derivations with the same score, one can use an arbitrary ordering over the partial derivations—for example, lexicographic order.

For a monotonic semiring, if

$$a_1 \otimes \bar{1} \otimes b_1 \leq a_2 \otimes \bar{1} \otimes b_2$$

then for all x,

$$a_1 \otimes x \otimes b_1 \leq a_2 \otimes x \otimes b_2$$

Thus, which term of Equation (21) is greatest does not depend on V. Total ordering implies that there is a unique greatest term. From Equation (17), only the greatest term of Equation (21) appears in the result, meaning that $F_{\neg X}(x) = a_j \otimes x \otimes b_j$ for some j. Our algorithm for extremal semirings identifies this greatest term and retains it as the outside value. Algorithm 4 represents outside values as pairs of semiring values to be combined on the left and right. For an outside value $Z(X)$, we use $Z(X).l$ to denote the first (or left) element of the pair, and $Z(X).r$ to denote the second (or right) element. (Including the term $\bar{1}$ in the products above is superfluous for semirings, because it is the multiplicative identity element. We retain it to indicate a placeholder for the inside values, and to generalize to settings where items may have different types, and an identity element of the same type as the inside value may be necessary.)

Algorithm 4 Outside Pass for Extremal Semiring

procedure OUTSIDEEXTREMAL
  $Z(G) \leftarrow (\bar{1}, \bar{1})$ for goal item G
  for Items X in reverse topological order **do**
    $Z(X) \leftarrow (\bar{0}, \bar{0})$
    for Rules $R = \frac{A_1, \ldots, A_n}{B}$, i such that $A_i = X$ **do**
      if $Z(X).l \otimes \bar{1} \otimes Z(X).r \leq$
          $Z(B).l \otimes (\bigotimes_{j=1}^{i-1} V(A_j)) \otimes \bar{1} \otimes (\bigotimes_{j=i+1}^{n} V(A_j)) \otimes Z(B).r$ **then**
        $Z(X).l \leftarrow Z(B).l \otimes \bigotimes_{j=1}^{i-1} V(A_j)$
        $Z(X).r \leftarrow (\bigotimes_{j=i+1}^{n} V(A_j)) \otimes Z(B).r$
    Set $F_{\neg X} = Z(X).l \otimes x \otimes Z(X).r$

The representation of $F_{\neg X}(x)$ that we have derived results in the following corollary of Theorem 1.

Corollary 2
Efficient outside computation is possible for any extremal semiring whose operations can be computed in time $O(g(|E|))$.

3.3 Sum of Linear Functions

As an example of a setting where efficient outside computation is possible even though the inside functions are not semiring operations, we consider the case of vectors as item weights. Components of these vectors correspond to latent variables or refined nonterminals in the latent variable parsing models of Matsuzaki, Miyao, and Tsujii (2005), Petrov et al. (2006), and Cohen et al. (2012).

To make this concrete, we take as our starting point the tensor formulation of the inside–outside algorithm given by Collins and Cohen (2012). Inside values for an

item are vectors, and the function for computing inside values bottom–up consists of applying a three-dimensional tensor $T^{a \to b\,c}$ specific to a CFG rule $a \to b\,c$ to two vectors representing the inside values for nonterminals b and c. The function for computing inside values takes two vectors as arguments, and returns a vector that is linear in each argument:

$$F_R(\mathbf{x}_a, \mathbf{x}_b)_k = \sum_{i,j} T^{a \to b\,c}_{i,j,k} x_{a,i}, x_{b,j}$$

If we project this function onto one of its arguments as shown in Equation (9), we obtain a linear function:

$$f_{p,i}(\mathbf{x}) = \mathbf{F}^{R_i}\,\mathbf{x} \tag{22}$$

where $\mathbf{F}^{R_i}$ is a matrix that can be computed from the rule tensor $T^{a \to b\,c}$ and the other argument of F_R. This implies that the outside function for an item is linear and can be expressed as a matrix-vector multiplication:

$$F_{\neg X}(\mathbf{x}) = \mathbf{Z}^{(\neg X)}\,\mathbf{x}$$

for some matrix $\mathbf{Z}^{(\neg X)}$. We now show this result by induction. From our composition rule in Equation (15):

$$F_{\neg X}(\mathbf{x}) = \sum_{\substack{R,i: \\ R=\frac{A_1,\ldots,A_n}{B} \\ A_i=X}} F_{\neg B}(F_R(V(A_1), \ldots, V(A_{i-1}), \mathbf{x}, V(A_{i+1}), \ldots, V(A_n))) \tag{23}$$

Using the linear projection of Equation (22):

$$= \sum_{\substack{R,i: \\ R=\frac{A_1,\ldots,A_n}{B} \\ A_i=X}} F_{\neg B}(\mathbf{F}^{R_i}\mathbf{x}) \tag{24}$$

Using the induction hypothesis:

$$= \sum_{\substack{R,i: \\ R=\frac{A_1,\ldots,A_n}{B} \\ A_i=X}} \mathbf{Z}^{(\neg B)}\,\mathbf{F}^{R_i}\,\mathbf{x} \tag{25}$$

$$= \mathbf{Z}^{(\neg X)}\,\mathbf{x} \tag{26}$$

Thus there exists a matrix $\mathbf{Z}^{(\neg X)}$ as desired.

The computation of the matrix $\mathbf{Z}^{(\neg X)}$ takes time constant in $|E|$, giving the following corollary of Theorem 1.

Corollary 3
Efficient outside computation is possible for any inside function consisting of a sum of linear functions.

This example does not fall into the semiring framework. The inside function cannot be expressed as a semiring product because the rule tensor and the vectors for inside values do not have the same type. It is also possible to allow different items to have

inside values consisting of vectors of different dimensionality, which therefore do not belong to a single semiring. Thus, the sum of linear functions provides a case where efficient outside computation is possible, despite the fact that the inside functions are not semiring operations, much less commutative or extremal semirings.

Matrix multiplication. The operations of matrix addition and matrix multiplication over $d \times d$ matrices of real numbers form a semiring in which efficient inside computation is possible. However, this semiring is non-commutative, and is also not extremal. Nevertheless, because the outside functions are linear, the sum of linear functions technique allows efficient outside computation.

For any semiring, including non-commutative semirings, we saw in Equation (16) that the outside function for an item can be represented as:

$$F_{\neg X}(x) = \bigoplus_p a_p \otimes x \otimes b_p$$

where p ranges over all paths from X to the goal item, and a_p and b_p are semiring values determined from the inside values along path p. In the case of matrix multiplication, this function cannot be represented as a single matrix multiplication. For example, if $\mathbf{V}$ is a matrix of rank one, the product of $\mathbf{V}$ with any matrix will have rank no greater than one, while the rank of $F_{\neg X}(\mathbf{V})$ may be as large as the number of terms in the sum. Thus, it is not possible to represent the outside value of an item as an element of the semiring used to define inside computation.

Nevertheless, $F_{\neg X}$ is a linear function from $\mathbb{R}^{d \times d}$ to $\mathbb{R}^{d \times d}$ having d^4 parameters. Consider the inside function for a rule R using matrix multiplication as the semiring product:

$$F_R(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \mathbf{A}\,\mathbf{B}\,\mathbf{C}$$

The projection of this function onto its second argument is:

$$f(\mathbf{B})_{i,j} = \sum_{k,\ell} a_{i,k} b_{k,\ell} c_{\ell,j}$$

which is a linear function with the d^4 parameters $\{a_{i,k} c_{\ell,j}\}_{i,j,k,\ell}$. In general, the projection onto any one argument has the form of Equation (22), repeated here:

$$f_{p,i}(\mathbf{x}) = \mathbf{F}^{R_i}\,\mathbf{x} \tag{27}$$

where $\mathbf{x}$ is a vector of dimensionality d^2 consisting of a flattened version of the $d \times d$ matrix for an inside value, and $\mathbf{F}^{R_i}$ is a matrix of size $d^2 \times d^2$. Matrix addition is equivalent to a sum of the flattened vectors. Thus, the semiring of matrix addition and matrix multiplication falls into the framework of a sum of linear functions, and efficient outside computation is possible using the procedure described above.

We emphasize that, although standard implementations of matrix muliplication are $O(d^3)$ time in the matrix dimension, the time is constant with respect to the size of the hypergraph $|E|$. Thus the function $g(|E|)$ in the statement of Theorem 1 is constant, and efficient outside computation is equivalent to time linear in $|E|$.

3.4 Superior Functions

We now give an example where efficient outside computation is not possible.

Table 1
Summary of results.

	Efficient Inside Possible	Best-first Possible
Efficient Outside Possible	commutative semirings sum of linear functions	extremal semirings
Efficient Outside Not Possible		general superior functions

Knuth's framework of a minimum of superior functions encompasses extremal semirings, as well as some cases outside the semiring framework. It allows not only efficient inside computation, but also efficient best-first search using a generalization of Dijkstra's algorithm (Nederhof 2003). However, efficient outside computation is not possible in general. Using Equation (10), the outside function can be represented as:

$$F_{\neg X}(x) = \min_p f_{p,n} \circ \cdots \circ f_{p,1}(x)$$

where p ranges over paths from X to the goal, and each $f_{p,i}$ is the inside function at rule i of path p, projected onto a single argument by fixing the values of the other argument to their inside values. This outside function is guaranteed to be a superior function, but may be arbitrarily complex. For example, even if each $f_{p,i}$ is linear, and therefore each composition of $f_{p,n} \circ \cdots \circ f_{p,1}$ is also linear, $F_{\neg X}(x)$ may be a piecewise linear function with an exponentially large number of pieces. Because there is no known way to perform the function composition and represent the result in constant time, efficient outside computation is not possible.

This implies that the conditions for efficient outside computation neither subsume nor are subsumed by the conditions for best first search, as summarized in Table 1.

4. Cycles

We now relax the assumption that the deduction system has no cycles. In the semiring framework, the total weight of an item X is defined by Goodman (1999) as:

$$\gamma(X) = \bigoplus_{D:X \in D} \text{weight}\,(D)\, C(X, D)$$

where $C(X, D)$ is an integer indicating the number of times that item X appears in derivation D, and the product weight $(D)\, C(X, D)$ indicates repeated addition with the semiring $\oplus$ operation. Inside values can be computed by solving a set of equations of the form of Equation (2). The equations may be linear, if an item can appear at most once as the antecedent of a rule (this is the case for unary chains in CFGs), or nonlinear, if an item can appear more than once (as can happen with CFGs with epsilon productions). Methods for solving such equations are discussed by Stolcke (1995) and Goodman (1999), with detailed complexity analysis by Etessami and Yannakakis (2009).

For commutative semirings, computing outside values once inside values are known involves solving a similar set of equations. The outside equations are always linear, because they have only one outside value on the right-hand side. For extremal semirings, derivations with cycles can always be discarded, as they have weight less

than the same derivation with the cycle removed, assuming that the inside value is well-defined. For the sum of linear functions, outside values can again be computed by solving a set of linear equations.

To summarize, for all cases discussed in this article where efficient outside computation is possible, outside computation with cycles is no more difficult than inside computation with cycles.

5. Conclusion

This article has aimed to provide a deeper understanding of the conditions under which efficient outside computation is possible by making three observations.

First, we give a very general condition for efficient outside computation stated in terms of function composition. Despite the emphasis in the literature on describing weighted deduction in terms of semirings, our general condition does not apply to all semirings, and can apply in situations that do not fall into the semiring framework.

Second, we identify a few more specific situations in which efficient outside computation is possible. Extremal semirings help explain why efficient outside computation is possible for the specific non-commutative semirings described by Goodman (1999), despite the fact that the general outside algorithm given by Goodman is not efficient. The sum of linear functions is a setting that is not a semiring but does allow efficient outside computation.

Third, we show that the conditions for efficient outside computation are incomparable to the conditions for efficient best-first search.

The bottom left cell of Table 1 is empty. It is an interesting open problem to consider whether this is an accident, which is to say, whether efficient outside computation is possible for all semirings. Resolving this problem would require either providing a general efficient algorithm that applies to all semirings, or providing a counterexample by means of a semiring such that outside computation can be used to solve a problem that is NP-complete or otherwise considered to be intractable.

Acknowledgments

We are grateful for feedback from Giorgio Satta, Daniel Štefankovič, Parker Riley, Shay Cohen, Esma Balkır, and the anonymous reviewers. This work was supported by National Science Foundation award IIS-1813823.

References

Alonso, Miguel A., David Cabrero, Eric de la Clergerie, and Manuel Vilares. 1999. Tabular algorithms for TAG parsing. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 150–157, Bergen. DOI: https://doi.org/10.3115/977035.977056

Baker, J. K. 1979. Trainable grammars for speech recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550.

Burden, Håkan and Peter Ljunglöf. 2005. Parsing linear context-free rewriting systems. In *9th International Workshop on Parsing Technologies (IWPT-05)*, pages 11–17, Vancouver. DOI: https://doi.org/10.3115/1654494.1654496

Cohen, Shay B., Robert J. Simmons, and Noah A. Smith. 2011. Products of weighted logic programs. In *Theory and Practice of Logic Programming*, 112–3:263–296. DOI: https://doi.org/10.1017/S1471068410000529

Cohen, Shay B., Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2012. Spectral learning of latent-variable PCFGs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 223–231, Jeju Island. DOI: https://doi.org/10.1017/S1471068410000529

Collins, Michael and Shay B. Cohen. 2012. Tensor decomposition for fast parsing with

latent-variable PCFGs. In *Advances in Neural Information Processing Systems 25*, pages 2519–2527, Curran Associates, Inc.

Cortes, Corinna, Mehryar Mohri, Ashish Rastogi, and Michael Riley. 2006. Efficient computation of the relative entropy of probabilistic automata. In *7th Latin American Symposium on Theoretical Informatics (LATIN 2006)*, volume 3887 of *Lecture Notes in Computer Science*, pages 323–336, Valdivia. DOI: https://doi.org/10.1007/11682462_32

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society*, 39(1):1–21. DOI: https://doi.org/10.1111/j.2517-6161.1977.tb01600.x

Eisner, Jason. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02)*, pages 1–8, Philadelphia, PA. DOI: https://doi.org/10.3115/1073083.1073085

Eisner, Jason. 2016. Inside-outside and forward-backward algorithms are just backprop. In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 1–17, Austin, TX. DOI: https://doi.org/10.18653/v1/W16-5901

Eisner, Jason and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Conference of the Association for Computational Linguistics (ACL-99)*, pages 457–464, College Park, MD. DOI: https://doi.org/10.3115/1034678.1034748

Etessami, Kousha and Mihalis Yannakakis. 2009. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the Association for Computing Machinery*, 56(1):1:1–66. DOI: https://doi.org/10.1145/1462153.1462154

Gimpel, Kevin and Noah A. Smith. 2009. Cube summing, approximate inference with non-local features, and dynamic programming without semirings. In *12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*, pages 318–326, Athens. DOI: https://doi.org/10.3115/1609067.1609102

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

Goodman, Joshua. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605. DOI: https://doi.org/10.1162/089120104185089 4

Hwa, Rebecca. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30(3):253–276.

Knuth, Donald E. 1977. A generalization of Dijkstra's algorithm. *Information Processing Letters*, 6(1):1–5.

Kuhlmann, Marco and Giorgio Satta. 2014. A new parsing algorithm for combinatory categorial grammar. *Transactions of the Association for Computational Linguistics*, 2:405–418. DOI: https://doi.org/10.1162/tacl_a_00192

Lari, K. and S. J. Young. 1990. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4:35–56.

Li, Zhifei and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*, pages 40–51, Singapore.

Lopez, Adam. 2009. Translation as weighted deduction. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 532–540, Athens. DOI: https://doi.org/10.3115/1609067.1609126, PMID 19815967

Matsuzaki, Takuya, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Conference of the Association for Computational Linguistics (ACL-05)*, pages 75–82, Ann Arbor, MI. DOI: https://doi.org/10.3115/1219840.1219850

Melamed, I. Dan, Giorgio Satta, and Ben Wellington. 2004. Generalized multitext grammars. In *Proceedings of the 42nd Annual Conference of the Association for Computational Linguistics (ACL-04)*, pages 661–668, Barcelona. DOI: https://doi.org/10.3115/1218955.1219039

Nederhof, M.-J. 2003. Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics*, 29(1):135–144. DOI: https://doi.org/10.1162/089120103321337467

Pereira, Fernando and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *30th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Newark, DE. DOI: https://doi.org/10.3115/981967.981984

Petrov, Slav, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney. DOI: https://doi.org/10.3115/1220175.1220230

Rumelhart, D. E., G. E. Hinton, and R. J. Williams. 1986. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 2. MIT Press, pages 318–362. DOI: https://doi.org/10.21236/ADA164453, PMID: 3159828

Shieber, Stuart M., Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1-2):3–36. DOI: https://doi.org/10.1016/0743-1066(95)00035-I

Sikkel, Klaas. 1997. *Parsing Schemata*. Springer Verlag, Berlin.

Stolcke, Andreas. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–202.

Vorobev, N. N. 1963. Extremal matrix algebra. *Proceedings of the USSR Academy of Sciences*, 152:24–27.