Weighted Deductive Parsing and Knuth's Algorithm

Mark-Jan Nederhof*
University of Groningen

We discuss weighted deductive parsing and consider the problem of finding the derivation with the lowest weight. We show that Knuth's generalization of Dijkstra's algorithm for the shortest-path problem offers a general method to solve this problem. Our approach is modular in the sense that Knuth's algorithm is formulated independently from the weighted deduction system.

1 Introduction

As for algorithms in general, there are significant advantages to specifying parsing algorithms in a modular way, i.e. as the combination of subalgorithms. First, modular specifications often allow simpler implementations. Secondly, if otherwise seemingly distinct types of parser are described in a modular way, the common parts can often be more readily identified, which helps to classify and analyse parsing algorithms.

In this paper we discuss a modular design for weighted deductive parsing, by distinguishing between a weighted deduction system on the one hand, which pertains to the choice of grammatical formalism and parsing strategy, and the algorithm that finds the derivation with the lowest weight on the other. The latter is Dijkstra's algorithm for the shortest-path problem (Dijkstra, 1959) as generalized by Knuth (1977) for a problem on grammars. It has been argued by e.g. Backhouse (2001) that this algorithm can be used to solve a wide range of problems on context-free grammars. A brief presentation of a very similar algorithm for weighted deductive parsing has been given before by Eisner (2000, Figure 3.5e).

Our presentation contrasts with that of Klein and Manning (2001), who offer an indivisible specification for a small collection of parsing strategies for weighted context-free grammars only, referring to a generalization of Dijkstra's algorithm to hypergraphs by Gallo et al. (1993).

This paper also addresses the efficiency of Knuth's algorithm for weighted deductive parsing, relative to the more commonly used algorithm by Viterbi.

2 Weighted Deductive Parsing

The use of deduction systems for specifying parsers has been proposed by Shieber, Schabes, and Pereira (1995) and Sikkel (1997). As already remarked by Goodman (1999), deduction systems can also be extended to manipulate weights.[1] Here we define such a

* Faculty of Arts, Humanities Computing, P.O. Box 716, NL-9700 AS Groningen, The Netherlands. E-mail: markjan@let.rug.nl. Secondary affiliation is the German Research Center for Artificial Intelligence (DFKI).

1 Weighted deduction is closely related to probabilistic logic, although the problem considered in this paper, viz. finding derivations with lowest weights, is different from typical problems in probabilistic logic. E.g. Frisch and Haddawy (1994) propose inference rules that manipulate logical formulas attached to *intervals* of probabilities, and the objective of deduction is to determine intervals that are as narrow as possible.

weighted deduction system for parsing as consisting of a finite set of inference rules of the form:

$$\frac{\begin{array}{c} x_1 : I_1 \\ x_2 : I_2 \\ \vdots \\ x_m : I_m \end{array}}{f(x_1, x_2, \ldots, x_m) : I_0} \quad \left\{ \begin{array}{l} c_1 \\ \vdots \\ c_p \end{array} \right.$$

where $m \geq 0$ and $p \geq 0$, and $I_0, I_1, \ldots, I_m$ are **items**, of which I_0 is the **consequent** and $I_1, \ldots, I_m$ are the **antecedents**, and $c_1, \ldots, c_p$ is a list of **side conditions** linking the inference rule to the grammar and the input string.[2] We assign unique variables $x_1, \ldots, x_m$ to each of the antecedents, and a **weight function** f, with $x_1, \ldots, x_m$ as arguments, to the consequent. This allows us to assign a weight to each occurrence of an (instantiated) item that we derive by an inference rule, by means of a function on the weights of the (instantiated) antecedents of that rule.

A weighted deduction system furthermore contains a set of **goal items**; like the inference rules, this set is parameterized by the grammar and the input. The objective of weighted deductive parsing is now to find the derivation of a goal item with the lowest weight. In this paper we assume that, for a given grammar and input string, each inference rule can be instantiated in a finite number of ways, which ensures that this problem can be solved under the constraints on the weight functions to be discussed in Sections 4 and 5.

Our examples will be restricted to context-free parsing, and include the deduction system for weighted bottom-up parsing in Figure 1, and that for weighted top-down parsing in Figure 2. The latter is very close to an extension of Earley's algorithm described by Lyon (1974). The side conditions refer to an input string $w = a_1 \cdots a_n$ and to a weighted context-free grammar (CFG) with a set of productions P, each of which has the form $(y : A \rightarrow \alpha)$, where y is a non-negative real-valued weight, A is a nonterminal, and α is a list of zero or more terminals or nonterminals. We assume the weight of a grammar derivation is given by the sum of the weights of the occurrences of productions therein.

Weights may be atomic entities, as in the deduction systems discussed above, where they are real-valued, but they may also be composed entities. E.g. Figure 3 presents an alternative form of weighted top-down parsing using pairs of values, following Stolcke (1995). The first value is the **forward** weight, i.e. the sum of weights of all productions that were encountered in the lowest-weighted derivation in the deduction system of an item $[A \rightarrow \alpha \bullet \beta, i, j]$. The second is the **inner** weight, i.e. it only considers the weight of the current production $A \rightarrow \alpha\beta$ plus the weights of productions in lowest-weighted grammar derivations for nonterminals in α. These inner weights are the same values as the weights in Figures 1 and 2. In fact, if we omit the forward weights, we obtain the deduction system in Figure 2.

Since forward weights pertain to larger parts of grammar derivations than the inner weights, they may be better suited to direct the search for the lowest-weighted complete grammar derivation. We assume a pair (z_1, x_1) is smaller than (z_2, x_2) if and only if $z_1 < z_2$ or $z_1 = z_2 \wedge x_1 < x_2$. (Tendeau (1997) has shown the general idea can also be applied to left-corner parsing.)

2 Note that we have no need for (explicit) axioms, since we allow inference rules to have zero antecedents.

$$\text{Initializer: } \frac{}{y : [B \rightarrow \bullet\, \gamma, j, j]} \left\{ \begin{array}{l} (y : B \rightarrow \gamma) \in P \\ 0 \leq j \leq n \end{array} \right.$$

$$\text{Scanner: } \frac{x_1 : [A \rightarrow \alpha \bullet a\beta, i, j]}{x_1 : [A \rightarrow \alpha a \bullet \beta, i, j+1]} \left\{ \begin{array}{l} (y_1 : A \rightarrow \alpha a \beta) \in P \\ 0 \leq i \leq j < n \\ a_{j+1} = a \end{array} \right.$$

$$\text{Completer: } \frac{\begin{array}{c} x_1 : [A \rightarrow \alpha \bullet B\beta, i, j] \\ x_2 : [B \rightarrow \gamma \bullet, j, k] \end{array}}{x_1 + x_2 : [A \rightarrow \alpha B \bullet \beta, i, k]} \left\{ \begin{array}{l} (y_1 : A \rightarrow \alpha B \beta) \in P \\ (y_2 : B \rightarrow \gamma) \in P \\ 0 \leq i \leq j \leq k \leq n \end{array} \right.$$

Goal items: $[S \rightarrow \gamma \bullet, 0, n]$ for any $(y : S \rightarrow \gamma) \in P$, where S is the start symbol

Figure 1
Weighted deduction system for bottom-up parsing

$$\text{Starter: } \frac{}{y : [S \rightarrow \bullet\, \gamma, 0, 0]} \left\{ \ (y : S \rightarrow \gamma) \in P, \text{ where } S \text{ is the start symbol} \right.$$

$$\text{Predictor: } \frac{x_1 : [A \rightarrow \alpha \bullet B\beta, i, j]}{y_2 : [B \rightarrow \bullet\, \gamma, j, j]} \left\{ \begin{array}{l} (y_1 : A \rightarrow \alpha B \beta) \in P \\ (y_2 : B \rightarrow \gamma) \in P \\ 0 \leq i \leq j \leq n \end{array} \right.$$

Scanner, completer and set of goal items are as in Figure 1.

Figure 2
Weighted deduction system for top-down parsing

$$\text{Starter: } \frac{}{(y, y) : [S \rightarrow \bullet\, \gamma, 0, 0]} \left\{ \ (y : S \rightarrow \gamma) \in P, \text{ where } S \text{ is the start symbol} \right.$$

$$\text{Scanner: } \frac{(z_1, x_1) : [A \rightarrow \alpha \bullet a\beta, i, j]}{(z_1, x_1) : [A \rightarrow \alpha a \bullet \beta, i, j+1]} \left\{ \begin{array}{l} (y_1 : A \rightarrow \alpha a \beta) \in P \\ 0 \leq i \leq j < n \\ a_{j+1} = a \end{array} \right.$$

$$\text{Predictor: } \frac{(z_1, x_1) : [A \rightarrow \alpha \bullet B\beta, i, j]}{(z_1 + y_2, y_2) : [B \rightarrow \bullet\, \gamma, j, j]} \left\{ \begin{array}{l} (y_1 : A \rightarrow \alpha B \beta) \in P \\ (y_2 : B \rightarrow \gamma) \in P \\ 0 \leq i \leq j \leq n \end{array} \right.$$

$$\text{Completer: } \frac{\begin{array}{c} (z_1, x_1) : [A \rightarrow \alpha \bullet B\beta, i, j] \\ (z_2, x_2) : [B \rightarrow \gamma \bullet, j, k] \end{array}}{(z_1 + x_2, x_1 + x_2) : [A \rightarrow \alpha B \bullet \beta, i, k]} \left\{ \begin{array}{l} (y_1 : A \rightarrow \alpha B \beta) \in P \\ (y_2 : B \rightarrow \gamma) \in P \\ 0 \leq i \leq j \leq k \leq n \end{array} \right.$$

Set of goal items is as in Figure 1.

Figure 3
Alternative weighted deduction system for top-down parsing

In order to link (weighted) deduction systems to literature to be discussed in Section 3, we point out that a deduction system having a grammar G in a certain formalism F and input string w in the side conditions can be seen as a construction c of a context-free grammar $c(G, w)$ out of grammar G and input w. The set of productions of $c(G, w)$ is obtained by instantiating the inference rules in all possible ways using productions from G and input positions pertaining to w. The consequent of such an instantiated inference rule then acts as the left-hand side of a production, and the (possibly empty) list of antecedents acts as its right-hand side. In the case of a weighted deduction system, the productions are associated with weight functions computing the weight of the left-hand side from the weights of the right-hand side nonterminals.

For example, if the input is $w = a_1 a_2 a_3$, and if there are two productions in the weighted CFG G of the form $(y_1 : A \to C\ B\ D), (y_2 : B \to E) \in P$, then from the completer in Figure 1 we may amongst others obtain a production $[A \to C\ B \bullet D, 0, 2] \to [A \to C \bullet B\ D, 0, 1]\ [B \to E \bullet, 1, 2]$, with associated weight function $f(x_1, x_2) = x_1 + x_2$, which states that if the production is used in a derivation, then the weights of the two subderivations should be added. The number of productions in $c(G, w)$ is determined by the number of ways we can instantiate inference rules, which in the case of Figure 1 is $\mathcal{O}(|G|^2 \cdot n^3)$, where $|G|$ is the size of G in terms of the total number of occurrences of terminals and nonterminals in productions.

If we assume, without loss of generality, that there is only one goal item, then this goal item becomes the start symbol of $c(G, w)$.[3] Since there are no terminals in $c(G, w)$, the grammar either generates the language $\{\epsilon\}$, containing only the empty string ϵ, or it generates the empty language; in the latter case, this indicates that w is not in the language generated by G.

Note that for all three examples above, the derivation with the lowest weight allowed by $c(G, w)$ encodes the derivation with the lowest weight allowed by G for w. Together with the dynamic programming algorithm to be discussed in the next section that finds the derivation with the lowest weight on the basis of $c(G, w)$, we obtain a modular approach to describing weighted parsers: one part of the description specifies how to construct grammar $c(G, w)$ out of grammar G and input w, and the second part specifies the dynamic programming algorithm to investigate $c(G, w)$.

Such a modular way of describing parsers in the unweighted case was already fully developed in the work by Lang (1974) and Billot and Lang (1989). Instead of a deduction system, they use a pushdown transducer to express a parsing strategy such as top-down parsing, left-corner parsing or LR parsing. Such a pushdown transducer can in the context of their work be regarded to specify a context-free grammar $c(G, w)$, given a context-free grammar G and an input string w. The second part of the description of the parser is a dynamic programming algorithm to actually construct $c(G, w)$ in polynomial time in the length of w.

This modular approach to describing parsing algorithms is also applicable to formalisms F other than context-free grammars. E.g. it was shown by Vijay-Shanker and Weir (1993) that tree-adjoining parsing can be realized by constructing a CFG $c(G, w)$ out of a tree-adjoining grammar G and an input string w. This can straightforwardly be generalized to weighted (in particular, stochastic) tree-adjoining grammars (Schabes, 1992).

It was shown by Boullier (2000) that F may furthermore be the formalism of range-concatenation grammars. Since the class of range-concatenation grammars generates exactly PTIME, this demonstrates the generality of the approach.[4]

Instead of string input, one may also consider input consisting of a finite automaton, along the lines of Bar-Hillel, Perles, and Shamir (1964); this can be trivially extended to

3 If there is more than one goal item, then a new symbol needs to be introduced as start symbol.
4 One may even consider formalisms F that generate languages beyond PTIME, but such applications of the approach would not necessarily be of practical value.

1. Let D be the empty set $\emptyset$.

2. Determine the set E and the function ν as follows:

   - E is the set of items $I_0 \notin D$ such that there is at least one inference rule from the deduction system that can be instantiated to a production of the form $I_0 \rightarrow I_1 \dots I_m$, for some $m \geq 0$, with weight function f, where $I_1, \dots, I_m \in D$.
   - For each such $I_0 \in E$, let $\nu(I_0)$ be the minimal weight $f(\mu(I_1), \dots, \mu(I_m))$ for all such instantiated inference rules.

3. If E is empty, then report failure and halt.

4. Choose an item $I \in E$ such that $\nu(I)$ is minimal.

5. Add I to D, and let $\mu(I) = \nu(I)$.

6. If I is a goal item, then output $\mu(I)$ and halt.

7. Repeat from step 2.

Figure 4
Knuth's generalization of Dijkstra's algorithm. Implicit are a weighted deduction system, a grammar G and an input w. For conditions on correctness, see Section 4.

the weighted case. That we restrict ourselves to string input in this paper is motivated by presentational considerations.

3 Knuth's Algorithm

The algorithm by Dijkstra (1959) effectively finds the shortest path from a distinguished source node in a weighted, directed graph to a distinguished target node. The underlying idea of the algorithm is that it suffices to investigate only the shortest paths from the source node to other nodes, since longer paths can never be extended to become shorter paths (weights of edges are assumed to be non-negative).

Knuth (1977) generalizes this algorithm to the problem of finding lowest-weighted derivations allowed by a CFG with weight functions, similar to those we have seen in the previous section. (The restrictions Knuth imposes on the weight functions will be discussed in the next section.) Again, the underlying idea of the algorithm is that it suffices to investigate only the lowest-weighted derivations of nonterminals.

The algorithm by Knuth is presented in Figure 4. We have taken the liberty to make some small changes to the formulation. The largest difference is that we have assumed that the CFG with weight functions on which the algorithm is applied has the form $c(G, w)$, obtained by instantiating the inference rules of a weighted deduction system for given grammar G and input w. Note however that $c(G, w)$ is *not* fully constructed before applying Knuth's algorithm, and only as much of it is accessed by the algorithm as is needed in its search for the lowest-weighted goal item.

In the algorithm, the set D contains items I for which the lowest overall weight has been found; this weight is given by $\mu(I)$. The set E contains items I_0 that can be derived in one step from items in D, but for which the lowest weight $\nu(I_0)$ found thus far may still exceed the lowest overall weight for I_0. In each iteration, it is established that the lowest weight $\nu(I)$ for an item I in E is the lowest overall weight for I, which justifies transferring I to D. The algorithm can be extended to output the derivation corresponding to the goal item with the lowest weight; this is fairly trivial and will not be discussed here.

A few remarks about the implementation of Knuth's algorithm are in order. First, instead of constructing E and ν anew at step 2 for each iteration, it may be more efficient to construct them only once, and revise them every time a new item I is added to D. This revision consists in removing I from E and combining it with existing items in D, as antecedents of inference rules, in order to find new items to be added to E and/or to update ν to assign lower values to items in E. Typically, E would be organized as a priority queue.

Secondly, practical implementations would maintain appropriate tables for indexing the items, in such a way that when a new item I is added to D, the lists of existing items in D together with which it matches the lists of antecedents of inference rules can be efficiently found. Since techniques for such kinds of indexing are well-established in the computer science literature, no further discussion is warranted here.

4 Conditions on the weight functions

A sufficient condition for Knuth's algorithm to correctly compute the derivations with the lowest weights is that the weight functions f are all **superior**, which means that they are monotone nondecreasing in each variable and that $f(x_1, \ldots, x_m) \geq \max(x_1, \ldots, x_m)$ for all possible values of $x_1, \ldots, x_m$. For this case, Knuth (1977) provides a short and elegant proof of correctness. Note that the weight functions in Figure 1 are all superior, so that correctness is guaranteed.

In the case of the top-down strategy from Figure 2 however, the weight functions are not all superior, since we have constant weight functions for the predictor, which may yield weights that are lower than their arguments. However, it is not difficult to show that Knuth's algorithm still correctly computes the derivations with the lowest weights, given that we have already established the correctness for the bottom-up case.

First note that items of the form $[B \rightarrow \bullet\ \gamma, j, j]$, which are introduced by the initializer in the bottom-up case, can be introduced by the starter or the predictor in the top-down case; in the top-down case, these items are generally introduced later than in the bottom-up case. Secondly, note that such items can contribute to finding a goal item only if from $[B \rightarrow \bullet\ \gamma, j, j]$ we succeed in deriving an item $[B \rightarrow \gamma\ \bullet, j, k]$ that is either such that $B = S$, $j = 0$ and $k = n$, or such that there is an item $[A \rightarrow \alpha\ \bullet\ B\beta, i, j]$. In either case, the item $[B \rightarrow \bullet\ \gamma, j, j]$ can be introduced by the starter or predictor so that $[B \rightarrow \gamma\ \bullet, j, k]$ will be available to the algorithm if and when it is needed to determine the derivation with the lowest weight for $[S \rightarrow \gamma\ \bullet, 0, n]$ or $[A \rightarrow \alpha B\ \bullet\ \beta, i, k]$, respectively, which will then have a weight larger than or equal to that of $[B \rightarrow \bullet\ \gamma, j, j]$.

For the alternative top-down strategy from Figure 3, the proof of correctness is similar, but now depends for a large part on the additional forward weights, the first values in the pairs (z, x); note that the second values are the inner weights, i.e. the weights we already considered in Figures 1 and 2. An important observation is that if there are two derivations for the same item with weights (z_1, x_1) and (z_2, x_2), respectively, such that $z_1 < z_2$ and $x_1 > x_2$, then there must be a third derivation of that item with weight (z_1, x_2). This shows that no relevant inner weights are overlooked due to the ordering we imposed on pairs (z, x).

Since Figures 1 thru 3 are merely examples to illustrate the possibilities of deduction systems and Knuth's algorithm, we do not provide full proofs of correctness.

5 Viterbi's Algorithm

This section places Knuth's algorithm in the context of a more commonly used alternative. This algorithm is applicable on a weighted deduction system if a simple partial order on items exists that is such that the antecedents of an inference rule are always strictly smaller than the consequent. When this is the case, we may treat items from small to large to compute their lowest weights. There are no constraints on the weight functions other than that they should be monotone nondecreasing.

The algorithm by Viterbi (1967) may be the earliest that operates by this principle. The partial order is based on the linear order given by a string of input symbols. In this paper we will let the name 'Viterbi's algorithm' refer to the general type of algorithm to search for the derivation with the lowest weight given a deduction system, a grammar, an input string, and a partial order on items consistent with the inference rules in the sense given above.[5]

Another example of an algorithm that can be seen as an instance of Viterbi's algorithm was presented by Jelinek, Lafferty, and Mercer (1992). This algorithm is essentially CYK parsing (Aho and Ullman, 1972) extended to handle weights (in particular, probabilities). The partial order on items is based on the sizes of their spans, i.e. the number of input symbols that the items cover. Weights of items with smaller spans are computed before the weights of those with larger spans.

In cases where a simple *a priori* order on items is not available but derivations are guaranteed to be acyclic, one may first determine a topological sorting of the complete set of derivable items and then compute the weights based on that order, following Martelli and Montanari (1978).

A special situation arises when a deduction system is such that inference rules allow cyclic dependencies within certain subsets of items but dependencies between these subsets represent a partial order. One may then combine the two algorithms: Knuth's (or Dijkstra's) algorithm is used within each subset and Viterbi's algorithm is used to relate items in distinct subsets. This is exemplified by Bouloutas, Hart, and Schwartz (1991).

In cases where both Knuth's algorithm and Viterbi's algorithm are applicable, the main difference between the two is that Knuth's algorithm may halt as soon as the lowest weight for a goal item is found, and no items with larger weights than that goal item need to be treated, whereas Viterbi's algorithm treats *all* derivable items. This suggests that Knuth's algorithm may be more efficient than Viterbi's. However, the worst-case time complexity of Knuth's algorithm involves an additional factor due to the maintenance of the priority queue. Following Cormen, Leiserson, and Rivest (1990), this factor is $\mathcal{O}(\log(\|c(G,w)\|))$, where $\|c(G,w)\|$ is the number of nonterminals in $c(G,w)$, which is an upper bound on the number of elements on the priority queue at any given time. Furthermore, there are observations by e.g. Chitrao and Grishman (1990), Tjong Kim Sang (1998, Sections 3.1 and 3.4) and van Noord et al. (1999, Section 3.9) that suggest that the apparent advantage of Knuth's algorithm does not necessarily lead to significantly lower time costs in practice.

In particular, consider deduction systems with items associated with spans as e.g. that in Figure 1, where the span of the consequent of an inference rule is the concatenation of the spans of the antecedents. If weights of individual productions in G differ only slightly, as is often the case in practice, then different derivations for an item have only

5 Note that some authors let the term 'Viterbi algorithm' refer to *any* algorithm that computes the 'Viterbi parse', i.e. the parse with the lowest weight or highest probability.

slightly different weights, and the lowest such weight for a certain item is roughly proportional to the size of its span. This suggests that Knuth's algorithm treats most items with smaller spans before any item with a larger span is treated, and since goal items typically have the maximal span, covering the complete input, there are few derivable items at all that are not treated before any goal item is found.

6 Conclusions

We have shown how a general weighted parser can be specified in two parts, the first being a weighted deduction system, and the second being Knuth's algorithm (or possibly Viterbi's algorithm where applicable). Such modular specifications have clear theoretical and practical advantages over indivisible specifications. For example, we may identify common aspects of otherwise seemingly distinct types of parser. Furthermore, modular specifications allow simpler implementations.

We have also identified close connections between our approach to specifying weighted parsers and well-established theory of grammars and parsing.

How the efficiency of Knuth's algorithm relates to that of Viterbi's algorithm in a practical setting is still to be investigated.

Acknowledgments

The author is supported by the Royal Netherlands Academy of Arts and Sciences. I am grateful to Gertjan van Noord, Giorgio Satta, Khalil Sima'an, Frederic Tendeau, and the anonymous referees for valuable comments.

References

Aho, Alfred V. and Jeffrey D. Ullman. 1972. *Parsing*, volume 1 of *The Theory of Parsing, Translation and Compiling*. Prentice-Hall.

Backhouse, Roland. 2001. Fusion on languages. In *10th European Symposium on Programming*, volume 2028 of *Lecture Notes in Computer Science*, pages 107–121. Springer-Verlag, April.

Bar-Hillel, Y., M. Perles, and E. Shamir. 1964. On formal properties of simple phrase structure grammars. In Y. Bar-Hillel, editor, *Language and Information: Selected Essays on their Theory and Application*. Addison-Wesley, chapter 9, pages 116–150.

Billot, Sylvie and Bernard Lang. 1989. The structure of shared forests in ambiguous parsing. In *27th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 143–151, Vancouver, British Columbia, Canada, June.

Boullier, Pierre. 2000. Range Concatenation Grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies*, pages 53–64, Trento, Italy, February.

Bouloutas, A., G.W. Hart, and M. Schwartz. 1991. Two extensions of the Viterbi algorithm. *IEEE Transactions on Information Theory*, 37(2):430–436, March.

Chitrao, Mahesh V. and Ralph Grishman. 1990. Statistical parsing of messages. In *Speech and Natural Language, Proceedings*, pages 263–266, Hidden Valley, Pennsylvania, June.

Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest. 1990. *Introduction to Algorithms*. The MIT Press.

Dijkstra, E.W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

Eisner, Jason. 2000. Bilexical grammars and their cubic-time parsing algorithms. In H. Bunt and A. Nijholt, editors, *Advances in Probabilistic and other Parsing Technologies*. Kluwer Academic Publishers, chapter 3, pages 29–61.

Frisch, Alan M. and Peter Haddawy. 1994. Anytime deduction for probabilistic logic. *Artificial Intelligence*, 69:93–122.

Gallo, Giorgio, Giustino Longo, Stefano Pallottino, and Sang Nguyen. 1993. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42:177–201.

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

Jelinek, F., J.D. Lafferty, and R.L. Mercer. 1992. Basic methods of probabilistic context free grammars. In P. Laface and R. De Mori, editors, *Speech Recognition and*

Understanding — Recent Advances, Trends and Applications. Springer-Verlag, pages 345–360.

Klein, Dan and Christopher D. Manning. 2001. Parsing and hypergraphs. In *Proceedings of the Seventh International Workshop on Parsing Technologies*, pages 123–134, Beijing, China, October.

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

Lang, Bernard. 1974. Deterministic techniques for efficient non-deterministic parsers. In *Automata, Languages and Programming, 2nd Colloquium*, volume 14 of *Lecture Notes in Computer Science*, pages 255–269, Saarbrücken. Springer-Verlag.

Lyon, Gordon. 1974. Syntax-directed least-errors analysis for context-free languages: A practical approach. *Communications of the ACM*, 17(1):3–14, January.

Martelli, Alberto and Ugo Montanari. 1978. Optimizing decision trees through heuristically guided search. *Communications of the ACM*, 21(12):1025–1039, December.

Schabes, Yves. 1992. Stochastic lexicalized tree-adjoining grammars. In *Proc. of the fifteenth International Conference on Computational Linguistics*, volume 2, pages 426–432, Nantes, August.

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

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

Stolcke, Andreas. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):167–201.

Tendeau, Frédéric. 1997. *Analyse syntaxique et sémantique avec évaluation d'attributs dans un demi-anneau*. Ph.D. thesis, University of Orléans.

Tjong Kim Sang, Erik F. 1998. *Machine Learning of Phonotactics*. Ph.D. thesis, University of Groningen.

van Noord, Gertjan, Gosse Bouma, Rob Koeling, and Mark-Jan Nederhof. 1999. Robust grammatical analysis for spoken dialogue systems. *Natural Language Engineering*, 5(1):45–93.

Vijay-Shanker, K. and David J. Weir. 1993. The use of shared forests in tree adjoining grammar parsing. In *Sixth Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference*, pages 384–393, Utrecht, The Netherlands, April.

Viterbi, Andrew J. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13(2):260–269, April.