# Computational Complexity of Probabilistic Disambiguation

*NP-Completeness Results for Parsing Problems That Arise in Speech and Language Processing Applications*

KHALIL SIMA'AN
*Institute for Logic, Language and Computation (ILLC), University of Amsterdam, Room B.234, Nieuwe Achtergracht 166, 1018 WV Amsterdam, The Netherlands*
*E-mail: khalil.simaan@hum.uva.nl*

**Abstract.** Recent models of natural language processing employ statistical reasoning for dealing with the ambiguity of formal grammars. In this approach, statistics, concerning the various linguistic phenomena of interest, are gathered from actual linguistic data and used to estimate the probabilities of the various entities that are generated by a given grammar, e.g., derivations, parse-trees and sentences. The extension of grammars with probabilities makes it possible to state ambiguity resolution as a constrained optimization formula, which aims at maximizing the probability of some entity that the grammar generates given the input (e.g., maximum probability parse-tree given some input sentence). The implementation of these optimization formulae in efficient algorithms, however, does not always proceed smoothly. In this paper, we address the computational complexity of ambiguity resolution under various kinds of probabilistic models. We provide proofs that some, frequently occurring problems of ambiguity resolution are NP-complete. These problems are encountered in various applications, e.g., language understanding for text- and speech-based applications. Assuming the common model of computation, this result implies that, for many existing probabilistic models it is not possible to devise tractable algorithms for solving these optimization problems.

**JEL codes:** D24, L60, 047

**Key words:** computational complexity, formal grammars, NP-Completeness, probabilistic ambiguity resolution

**Abbreviations:** NLP – natural language processing

## 1. Introduction

No matter what its state of knowledge is, a so called *performance model* (Chomsky, 1965) of syntactic processing is expected to model, *as accurately as possible*, the *input–output* language processing behavior of a human. One particular "input-output property" of the human language processing system is of interest here: a human tends to associate a single, "most plausible" analysis with every utterance that she encounters. Remarkably, this is in sheer contrast with the highly ambiguous formal linguistic grammars of natural languages. In this paper, we take a closer look at some *probabilistic* performance models of syntactic processing, which are aimed at resolving grammatical ambiguity. We inspect the computational complexity that accompanies the application of these models to various problems of syntactic ambiguity resolution. Before we enter the computational complexity is-

*Grammars* **5:** 125–151, 2002.

sues, however, we start out this paper with a brief overview of the general approach to probabilistic ambiguity resolution in Natural Language Processing (NLP).

In the field of NLP, it is widely believed that the solution to syntactic grammatical *ambiguity* lies in the modeling of the extra-linguistic knowledge, e.g., world-knowledge, which plays a pivotal role in human language processing. Because of their vastness and dynamic nature, the major extra-linguistic factors (e.g., world-knowledge or knowledge on cultural preferences) seem to be beyond complete modeling. Therefore, it seems inevitable that a model of syntactic ambiguity resolution must approximate these extra-syntactic factors by suitable means that allow for the optimization of its behavior given the current state of knowledge.

Recently, empirical performance models of syntactic processing started to implement ambiguity resolution through the projection of *probabilistic grammars* from so called tree-banks[1] (e.g., Scha, 1990; Bod, 1992; Pereira and Schabes, 1992; Jelinek et al., 1994; Magerman, 1995; Collins, 1997; Charniak, 1999). In the context of this paper we are not so much interested in the different methods of projecting probabilistic grammars as much as in the computational aspects of the disambiguation problems under these probabilistic models.

Informally speaking, a probabilistic grammar (e.g., Salomaa, 1969) attaches to every rewrite-event (or grammar rule) $r$ a *conditional probability* $P(r|C)$, expressing the probability of applying the rule given some context $C$. The probabilities are essentially "statistics" over syntactic phenomena in the tree-bank; they can be seen to represent the accumulations of the various extra-syntactic factors. Beyond the probabilistic grammar, a probabilistic language model stipulates how the probability of a *derivation* of the grammar is estimated from the conditional probabilities of the individual rewrite-events. Consequently, the probabilities of parse-trees and sentences are defined in terms of the probabilities of the derivations that generate them (Section 3 provides example models). In other words, a probabilistic model defines probability functions over derivations, parse-trees and sentences of the probabilistic grammar.

In this probabilistic view, the problem of syntactic ambiguity resolution for language utterances can be expressed as an optimization problem. Given an input utterance $U$, the goal is to find the parse-tree $T^*$ with the highest probability (most

[1] A tree-bank consists of a bag (or multi-set or distribution) of parse-trees (lexicalized with terminal symbols) from some source grammar. In natural language processing, a tree-bank is usually obtained by collecting utterances from some domain of language use "at random" and manually annotating them with parse-trees. This way, it can be assumed (to some extent) that the tree-bank provides a representative sample of that domain. Most existing tree-banks (e.g., Marcus et al., 1993) are annotated with Phrase-Structure linguistic notions.

probable parse) under the optimization formula:[2]

$$T^* = \arg\max_{T \in \mathcal{T}} P(T|U) = \arg\max_{T \in \mathcal{T}} \frac{P(T,U)}{P(U)} = \arg\max_{T \in \mathcal{T}} P(T,U) \quad (1)$$

where $\mathcal{T}$ is the set of parse-trees defined by the grammar, i.e.,, the tree-language of the grammar. Hence, $T^*$ is the most probable parse-tree which co-occurs with utterance $U$. A major question in applying probabilistic models for syntactic disambiguation is *whether it is possible* in any way, and then *how*, to transform optimization formula 1 into *efficient* algorithms?

In Computational Complexity theory, the departure point in addressing these questions lies in classifying the problems according to the time-complexities of the *most efficient* algorithms that can be devised for solving these problems. A problem that does not have any solutions of some desirable time-complexity constitutes a source of inconvenience and demands special treatment.

The present paper provides proofs that many of the common problems of probabilistic disambiguation, under various existing probabilistic models, belong to a class of problems for which we do not know whether we can devise *deterministic polynomial-time algorithms*. In fact, there is substantial evidence (see Garey and Johnson, 1981; Barton et al., 1987) that the problems that belong to this class, the *NP-complete class*, do not have such algorithms. For NP-complete problems, the only known deterministic algorithms have *exponential-time* complexity. This is inconvenient since exponential-time algorithms imply a serious limitation on the kinds and sizes of applications for which probabilistic models can be applied.

Each of the problems considered in this paper involves probabilistic disambiguation of some kind of input under performance models that are based either on Stochastic Context-Free Grammars (SCFGs) (e.g., Jelinek et al., 1990; Black et al., 1993; Charniak, 1996) or on Stochastic Tree-Substitution Grammars (STSGs) (e.g., Bod, 1992, 1995a; Sekine and Grishman, 1995; Sima'an, 2000). The results of the present proofs apply also to the various Stochastic versions of Tree Adjoining-Grammars (STAG) (Joshi, 1985; Resnik, 1992; Schabes, 1992; Schabes and Waters, 1993). It is noteworthy that the present proofs concern disambiguation problems that arise within various, actually existing applications that range from those that involve parsing and interpretation of text to those that involve speech-understanding and information retrieval. We will elaborate on these applications during the presentation of the problems, and, where possible, we also provide pointers to related literature.

This paper is aimed at readership both from Computer Science and Linguistics. Therefore, it includes an intuitive (rather than formal) brief overview of the notions of intractability and NP-completeness (Section 2), and formalizations of the

[2] The meaning of the expression $\arg\max_{x \in X} f(x)$ is "the element $x \in X$ for which $f(x)$ is maximal".

abstract forms of some of the existing models of ambiguity resolution in language processing (Section 3). Section 4 states the disambiguation problems in question. Section 5 provides the proofs of NP-completeness. Finally, Section 6 discusses the informal conclusions of this study and provides pointers to existing research toward approximate, efficient solutions to some of these problems.

## 2. Intractability and NP-Completeness

This section provides a short, informal overview of the notions of intractability and NP-completeness. Our aim is to provide the basic terminology and explain the intuitive side of intractability, thereby also highlighting some of its limitations. The reader that is interested in further details and formalizations of intractability and NP-completeness are referred to (e.g., Hopcroft and Ullman, 1979; Garey and Johnson, 1981; Lewis and Papadimitriou, 1981; Davis and Weyuker, 1983; Barton et al., 1987). This section also serves as a reference to the 3SAT problem, used in the sequel for our proofs of NP-completeness of disambiguation problems.

### 2.1. WHAT IS A DECISION PROBLEM?

The notion in focus in this section is that of a *tractable decision problem*. Informally speaking, a problem is a pair: a generic instance, stating the formal devices and components involved in the problem, and a question asked in terms of the generic instance. A *decision problem* is a problem where the question can have only one of two possible answers: *Yes* or *No*. For example, the well known 3SAT (3-satisfiability) problem[3] is stated as follows:

INSTANCE: A Boolean formula in 3-conjunctive normal form (3CNF) over the variables $u_1, \ldots, u_n$.

QUESTION: Is the formula in INSTANCE *satisfiable*? i.e., is there an assignment of values **true** (**T**) or **false** (**F**) to the Boolean variables $u_1, \ldots, u_n$ such that the given formula is true?

In the sequel, we refer to the (generic) instance of 3SAT with the name INS, and we use the following generic form to express the formula in INS:

$$(d_{11} \vee d_{12} \vee d_{13}) \wedge (d_{21} \vee d_{22} \vee d_{23}) \wedge \cdots \wedge (d_{m1} \vee d_{m2} \vee d_{m3}),$$

where $m \geqslant 1$ and $d_{ij}$ is a literal[4] over $\{u_1, \ldots, u_n\}$, for all $1 \leqslant i \leqslant m$ and all $1 \leqslant j \leqslant 3$. In some cases it is convenient to express the same formula using the notation $C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where $C_i$ represents $(d_{i1} \vee d_{i2} \vee d_{i3})$, for all $1 \leqslant i \leqslant m$.

[3] The 3SAT problem is a restriction of the more general satisfiability problem SAT which is the first problem proven to be NP-complete (known as Cook's theorem).

[4] A literal is a Boolean variable (e.g., $u_k$), or the negation of a Boolean variable (e.g., $\overline{u_k}$).

Decision problems are particularly convenient for complexity studies mainly because of the natural correspondence between them and the formal object called "language"; the formal form of a decision problem is usually stated in terms of a language and a question concerning set-membership. The size or length of an instance of a decision problem is the main variable in any measure of the time-complexity of the algorithmic solutions to the problem (in case these solutions exist). Roughly speaking, this length is measured with respect to some *reasonable encoding* (deterministic polynomial-time computable) from each instance of the decision problem to a string in the corresponding language. In order not to complicate the discussion more than necessary, we follow common practice, as explained in Garey and Johnson (1981), and assume measures of length that are more "natural" to the decision problem at hand (knowing that it is at most a polynomial-time cost to transform an instance to a string in the language corresponding to the decision problem). For 3SAT, for example, the length of an instance is $3m+(m-1)+2m$, i.e., the number of symbols in the formula (not counting the parentheses) is linear in the number of conjuncts $m$.

## 2.2. TRACTABLE PROBLEMS, CLASS P AND CLASS NP

Roughly speaking, the theory of intractability and NP-completeness deals with the question whether a given problem has a general solution that is "computationally feasible". In other words:

> For every instance of the problem and for every input that is fed to that instance, of length $n \geqslant 1$: is there a (deterministic) algorithmic solution, which computes the answer in a number of computation steps that is proportional to a "cheap" function in $n$?

The problem with defining the term "cheap" lies in finding a borderline between those functions that can be considered expensive and those that can be considered cheap. A first borderline that has been drawn by a widely accepted thesis (Cook-Karp) is between *polynomials* and *exponentials*. Problems for which there is a deterministic polynomial-time solution (i.e.,, can be solved by a Deterministic Turing Machine — DTM — in Polynomial-time) are called *tractable*. Other problems (in the set of problems that can be solved by Turing Machines) for which there are only deterministic exponential-time solutions (Turing Machines) are called *intractable*. The motivation behind the Cook-Karp discrimination between polynomials and exponentials is the difference in *rate of growth* between these two families. Generally speaking, exponentials tend to grow much faster than polynomials. Strong support for the Cook-Karp thesis came from practice, but here we will not go into the discussion concerning the stability of this thesis (see the discussions in e.g., Garey and Johnson, 1981; Barton et al., 1987).

In any case, the tractable decision problems, i.e.,, those that have a deterministic polynomial-time algorithmic solution (i.e., a polynomial-time DTM), are referred

to with the term *class P* problems. All other decision problems,[5] that are intractable, are referred to with the term NP-hard problems (see below for the reason for this terminology).

Besides the problems that can be solved in polynomial-time by a deterministic algorithmic solution, there exist problems that are solvable in polynomial-time *provided that the algorithmic solution has access to an oracle, which is able to guess the right computation-steps without extra cost* (i.e., these problems are solvable by a so called Non-Deterministic Turing Machine (NDTM) in polynomial-time). Clearly, every problem in class P is found among these so called Non-deterministic Polynomial-time solvable problems, also called class NP problems (i.e., $P \subseteq NP$). The question is, of course, are there more problems in class NP than in class P, i.e., $P \subset NP$?

### 2.3. NP-COMPLETE PROBLEMS

Empirical clues to the hypothesis $P \neq NP$ is embodied by the discovery of many practical and theoretical problems that are known to be in class NP but for which nobody knows how to devise *deterministic polynomial-time* algorithmic solutions; these problems are in class NP but are *not known* to be in class P. This set of problems is called the class of NP-complete problems. NP-complete problems are the hardest problems in class NP.

Further of interest here is the class of *NP-hard* problems. This class consists of those problems that are *at least as hard as any problem that is in NP*, i.e., an NP-hard decision problem is one that is at least as hard as those that can be solved by an NDTM in polynomial-time.

To prove that a given problem $L$ is NP-hard, it is sufficient[6] to show that another problem that is already known to be NP-complete is *deterministic polynomial-time reducible* to $L$. This is done by providing a polynomial-time reduction (i.e., transform) from the NP-complete problem to problem $L$. Such a reduction shows how every instance of the NP-complete problem can be transformed into an instance of problem $L$. Naturally, the reduction must be *answer-preserving*, i.e., for every instance of the NP-complete problem and for every possible input, the instance's answer is *Yes* to that input iff the $L$-instance's (resulting from the reduction) answer is also *Yes* to the transformed-form of the input. Note that the reducibility relation between problems is a transitive relation.

Practically speaking, once we lay our hands on one NP-complete problem, we can prove other problems to be NP-hard. A problem that has been proven to be NP-complete is the 3SAT problem stated above. Therefore 3SAT can serve us in proving other problems to be NP-hard. To prove that a new problem is NP-

[5] In the class of problems solved by Turing Machines.

[6] The fact that every instance of an NP-complete problem $T$ can be reduced into an instance of $L$ in deterministic polynomial-time implies that $L$ is at least as hard as $T$ – if $L$ would be solvable in deterministic polynomial-time then $T$ would also be.

completeness, however, one needs to prove that it is NP-hard and that it is in class NP.

In this work the focus is on optimization problems rather than decision problems. In general, it is possible to derive from every optimization problem a decision problem that is (at most) as hard as the optimization problem (Garey and Johnson, 1981). Therefore, the proof of NP-completeness of an optimization problem goes through proving that its decision counterpart is NP-complete. We will exemplify how an optimization problem is translated into a decision problem in Section 4 which states the problems that this paper addresses: probabilistic ambiguity resolution. But first, however, we consider the formal probabilistic models that play a role in these problems.

### 2.4. GENERAL NOTATION

In the sequel we employ the notation $x_1^n$ for an ordered sequence of entities $x_1, \ldots, x_n$ (and also for the concatenation of symbols $x_1 \ldots x_n$), for any kind of entity $x_i, 1 \leqslant i \leqslant n$. We also use the symbols **T** and **F** to represent respectively the Boolean values true and false.

## 3. Probabilistic Models of Ambiguity Resolution

The generic devices that are involved in the disambiguation problems considered in this paper are of three kinds: Stochastic Finite State Automata (SFSAs), Stochastic Context-Free Grammars (SCFGs) and Stochastic Tree-Substitution Grammars (STSGs). Next we provide the formal definitions of these devices, and specify the probability formulae common for most existing models of ambiguity resolution. It is relevant here to mention that the proofs of NP-hardness in this paper hold for any "weighted" forms of the above grammars in general. However, in this presentation we stick to "probabilistic/stochastic" versions because of the current wide interest in applying these specific versions to NLP tasks. Moreover, this allows us to specify some general aspects of existing probabilistic models, which might be of interest for the general readership.

Throughout the paper we assume that all automata and grammars are $\epsilon$-free (i.e., do not allow empty-rewritings) and cycle-free.

### 3.1. STOCHASTIC FINITE STATE AUTOMATA

**Stochastic Finite State Automaton (SFSA):** *An SFSA is a six-tuple* $\langle Q, \Gamma, T, s, f, P\rangle$ *where* $Q$, $\Gamma$ *and* $T$ *are finite sets of, respectively, symbols, states and transitions; a transition is a triple* $\langle s1, s2, w\rangle$, *where* $s1, s2 \in \Gamma$ *and* $w \in Q$ *(called the label of the transition). The special states* $s$ *and* $f$ *(both in* $\Gamma$*) are respectively the start and target states. Finally,* $P : T \rightarrow (0, 1]$ *is a probability function which fulfills the equation* $\forall l \in \Gamma$*:* $\sum_{\langle l,r,w\rangle \in T} P(\langle l, r, w\rangle) = 1$.

Two transitions $\langle a, b1, w\rangle$ and $\langle b2, c, v\rangle$ are called *connected* iff $b1 = b2$. An ordered sequence of one or more transitions $t_1, \cdots, t_n$, with $t_i = \langle l_i, r_i, w_i\rangle$ for all $i$, is called a *path* iff: (1) $l_1 = s$, (2) $r_n = f$ and (3) for all $1 \leqslant i < n$ holds $t_i$ and $t_{i+1}$ are connected. For any path $\langle l_1, r_1, w_1\rangle, \cdots, \langle l_m, r_m, w_m\rangle$, the concatenation of the labels of the transitions, i.e., $w_1 \cdots w_m$, is called a *sentence* generated by the SFSA through that path[7]. The set of all paths that generate the same sentence $X$ under a given SFSA is denoted by $Paths(X)$. In the present context, two probability definitions are relevant:

**Path probability:** the probability[8] of a path $t_1^n = t_1, \cdots, t_n$ is approximated[9] by the formula $P(t_1^n) = P(t_1)\prod_{i=2}^{n} P(t_i|t_1, \cdots, t_{i-1}) \approx \prod_{i=1}^{n} P(t_i)$.

**Sentence probability:** the probability of a sequence of symbols $w_1^n = w_1 \cdots w_n$ from $Q^+$ is given by:

$$P(w_1^n) = \sum_{path \in Paths(w_1^n)} P(path). \tag{2}$$

Note that if the set $Paths(w_1^n)$ is empty $P(w_1^n) = 0$ by definition.

Further definitions pertaining to methods of probability estimation and probability computation under SFSAs originate from the well known Hidden-Markov models (HMM) (Rabiner and Juang, 1986). HMMs are commonly used in speech-recognition as speech and language models. HMMs also provide a general interface between speech-recognizers and more advanced probabilistic language models (e.g., based on linguistic knowledge): the output of the speech-recognizer is cast in an SFSA (over some lexicon), called a "word-lattice" or "word-graph" (Oeder and Ney, 1993), which is fed as input to the language model (e.g., a syntactic analyzer); usually, the output of the language model is the single sentence (among the sentences accepted by the SFSA) which constitutes the system's "best" guess of the spoken utterance. The formal version of the latter disambiguation task constitutes one of the problems which we prove NP-hard (under various probabilistic language models).

For the purposes of the present paper it is sufficient to consider a special case of SFSAs, which we refer to with the term "Simple FSA" (abbreviated $\mathcal{SIFSA}$).

[7] In this paper we will assume that all states in the SFSA are "found" on paths, i.e., (1) all states are reachable (i.e., through sequences of connected transitions) from the start state $s$, and (2) the final state $f$ is reachable from all other states.

[8] We will allow overloading of $P$ since it is a probability mass function over entities which are always clear from the context.

[9] Considering the path as the joint-event of the individual transitions, this formula assumes stochastic independence between the different transitions. In Markov models, this independence assumption is relaxed to a certain degree by conditioning the probability of every transition on a few of the transitions that precede it in the path, e.g., for a first-order Markov model $P(t_1^n) \approx P(t_1)\prod_{i=2}^{n} P(t_i|t_1)$. Although important for modeling various tasks, this Markovian conditioning of probabilities is of no impact on our complexity results.

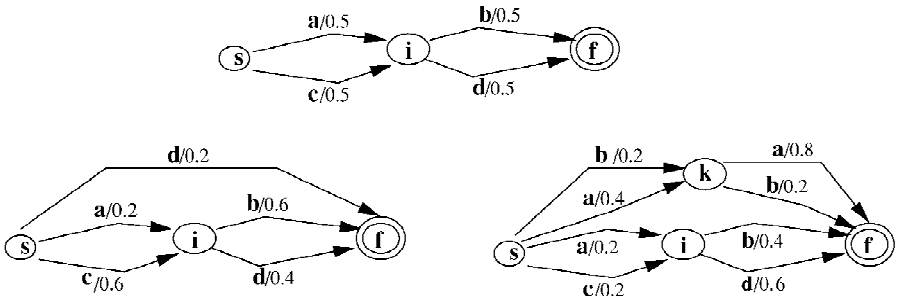


*Figure 1.* Upper: an $\mathcal{SIFSA}$. Lower: SFSA's which are not an $\mathcal{SIFSA}$.

To this end, we define the term "non-labeled transition": a non-labeled transition is the pair $\langle s_i, s_j \rangle$ obtained from any transition $\langle s_i, s_j, w \rangle \in T$. A *non-labeled path* consists of a sequence of connected non-labeled transitions starting from $s$ and ending at $f$, i.e., $\langle s, s_1 \rangle, \cdots, \langle s_n, f \rangle$.

**Simple SFSA:** *An $\mathcal{SIFSA}$ is an SFSA $\langle Q, \Gamma, T, s, f, P \rangle$ in which (1) all non-labeled paths constitute exactly the same sequence of connected non-labeled transitions and (2) the probabilities of transitions are uniform, i.e., $\forall l \in \Gamma$, $P(\langle l, r, w \rangle) = \frac{1}{|NEXT(l)|}$, where $NEXT(l) = \{r | \langle l, r, w \rangle \in T\}$, and $|Y|$ represents the cardinality of set $Y$.*

Figure 1 exhibits examples of $\mathcal{SIFSA}$*s* and other SFSAs. The definition of $\mathcal{SIFSA}$ implies the following notation for representing $\mathcal{SIFSA}$'s:

**States:** the states of an $\mathcal{SIFSA}$ can be mapped to *successive* positive natural numbers. We will represent the natural number that a state $s_i$ is mapped to by $N(s_i)$. For the start state $s$: $N(s) = 1$. For every other state $s_j$, if there is a transition $\langle s_i, s_j \rangle \in T$ with $N(s_i) = k$, then $N(s_j) = k + 1$.

**Sentences:** the set of sentences accepted by an $\mathcal{SIFSA}$ can be written as the product between sets $Q_1 \times \cdots \times Q_m$, where $Q_i \subseteq Q$, for all $1 \leqslant i \leqslant m$. Precisely, the set $Q_i$ consists of the labels of the transitions between the state $s_x$ for which $N(s_x) = i$ and the state $s_y$ for which $N(s_y) = i + 1$, $\forall 1 \leqslant i \leqslant m$.

Because in this paper we are mainly interested in the set of sentences $Q_1 \times \cdots \times Q_m$ accepted by an $\mathcal{SIFSA}$, we will be informal in referring to this set with the term "an $\mathcal{SIFSA}$" (under the understanding that we can readily construct an $\mathcal{SIFSA}$ for recognizing this set). Furthermore, we will use the notation $Q_1^m$ as shorthand for $Q_1 \times \cdots \times Q_m$.

$\mathcal{SIFSA}$*s* are often encountered in applications such as spelling-correction, under the assumption that typos do not add or remove white-space[10]. Clearly, because

[10] For every word containing a typo, it is assumed that a set of words are hypothesized from an external dictionary.

the proofs in this paper concern $\mathcal{SIFSAs}$, they also hold for SFSAs in general. We undertook the task of formalizing general SFSAs precisely because they constitute the most common device which plays a major role in practical applications of probabilistic models of ambiguity resolution (Figure 1).

### 3.2. STOCHASTIC TREE-SUBSTITUTION GRAMMARS

We define Stochastic Context-Free Grammars (SCFGs) (Salomaa, 1969; Jelinek et al., 1990) through the definition of the more general Stochastic Tree-Substitution Grammars (STSG) (Bod, 1992; Schabes, 1992).

Let $N$ and $T$ denote respectively the finite set of non-terminal symbols and terminal symbols.

**Elementary-tree:** *An elementary-tree is a tree-structure which fulfills: the non-leaf nodes are labeled with non-terminal symbols from $N$ and the leaf nodes are labeled with symbols from $N \cup T$.*

We use the functional notation $root(t)$ to represent the non-terminal label of the root node of the elementary-tree $t$.

**Stochastic Tree-Substitution Grammar (STSG)** *An STSG $G$ is a five tuple $\langle N, T, R, \mathcal{S}, P \rangle$, where the first three components are respectively the finite set of non-terminals, terminals and* elementary-trees. *The non-terminal $\mathcal{S} \in N$ is called the start-symbol. The function $P : R \rightarrow (0, 1]$ fulfills the following set of equations (constraints): $\forall A \in N : \sum_{\{t \in R : root(t) = A\}} P(t) = 1$.*

The term *partial parse-tree* is central in defining the various rewrite notions under STSGs. Every elementary-tree is a *partial parse-tree*. Further partial parse-trees are generated through partial derivations as defined next.

Given a partial parse-tree $T$, let node $\mu$ be the left-most leaf node in $T$ and let $\mu$ be labeled with the non-terminal $X$. A *derivation-step* from $\mu$ under STSG $G$ is a rewriting of non-terminal $X$ with an elementary-tree $t \in R$ which fulfills $root(t) = X$. The result of this rewrite step is just as in the well known algebraic "variable substitution": it results in a new partial parse-tree $T \circ t$ which consists of a duplicate of $T$ with a duplicate of elementary-tree $t$ substituted[11] for the duplicate of $\mu$. The operation of (left-most) *substitution*, denoted by $\circ$, distinguishes this kind of grammar. Hence "Stochastic Tree-Substitution Grammar".

A sequence of left-most substitutions $t_1 \circ \cdots \circ t_m$ is called a *partial derivation* of $G$. A partial-derivation generates a partial parse-tree under the interpretation $(\cdots (t_1 \circ t_2) \circ \cdots \circ t_m)$.

**Probability of a partial derivation:** *The probability of partial derivation $d$ is calculated by the formula $P(d) = \prod_{i=1}^{m} P(t_i)$.*

[11] In the sense that the duplicate of $\mu$ now is a non-leaf node dominating a duplicate of $t$.

In STSGs (in contrast with SCFGs), it is possible that various partial derivations generate one and the same partial parse-tree. This happens, for example, whenever there are elementary-trees $t_1, t_2, t_3, t_4 \in R$ such that $t_1 \circ t_2$ generates the same partial parse-tree as $t_3 \circ t_4$.

**Probability of a partial parse-tree:** *Let $der(T)$ denote the finite set of all partial derivations that generate partial parse-tree $T$. The probability of $T$ being generated is then defined by:*

$$P(T) = \sum_{d \in der(T)} P(d). \tag{3}$$

A partial derivation $d = t_1 \circ \cdots \circ t_m$ is also called a *derivation* iff it fulfills: (1) $root(t_1) = \mathcal{S}$, and (2) $d$ generates a partial parse-tree $T$ with leaf nodes labeled only with terminal symbols; in that case, $T$ is called a *parse-tree* and the ordered sequence of terminal symbols labeling the leaf nodes (left-to-right) of $T$ is called a *sentence* (generated by $d$).

In natural language processing, often the grammars are highly ambiguous such that one and the same sentence can be generated together with different parse-trees.

**Probability of a sentence:** *Let the set of parse-trees which are generated together with sentence $U$ under $G$ be denoted by $par(U)$ and the set of all derivations that generate $U$ be denoted by $der(U)$, then:*

$$P(U) = \sum_{T \in par(U)} P(T). \tag{4}$$

$$= \sum_{d \in der(U)} P(d). \tag{5}$$

Formulae (3) and (5) are quite important in the context of this paper. We will see that this kind of a definition (a sum over multiplications) plays a central role in all the optimization problems which we prove NP-hard.

#### 3.2.1. *Some convenient notation*

Whenever we do not wish to specify the sets $par(U)$ we may consider the probability of the joint occurrence of a parse-tree $T$ with sentence $U$, i.e., $P(T, U)$. This probability is defined to be zero if the sequence of terminal labels on the leaf nodes of $T$ is different from $U$. Under this definition we may write: $P(U) = \sum_T P_G(T, U)$, where the subscript $G$ implies that the summation ranges over all parse trees $T$ generated by STSG $G$. Similarly, the joint probability of a derivation $d$ with parse-tree $T$ (or sentence $U$) is defined to be zero if $d$ does not generate $T$ (respectively $U$) in the given STSG. Under this definition we may write

$P(U) = \sum_d P_G(d, U)$ and $P(T) = \sum_d P_G(d, T)$, where again the summation ranges over all derivations possible under STSG $G$.

### 3.3. STOCHASTIC CONTEXT-FREE GRAMMARS

Now that we defined STSGs, we can define Stochastic Context-Free Grammars (SCFGs) as a subclass thereof:

**Stochastic Context-Free Grammar:** *An SCFG $\langle N, T, R, \mathcal{S}, P\rangle$ is an STSG which fulfills: every elementary-tree $t \in R$ consists of a single "level" of non-leaf nodes, i.e., the root node of $t$ dominates directly all its leaf nodes In SCFG terminology, elementary-trees are also called productions or (rewrite-)rules.*

In contrast with general STSGs, in SCFGs there is a one-to-one mapping between derivations[12] and parse-trees. On the one hand, we will use SCFGs to show that this fact makes the syntactic ambiguity resolution of an input sentence easier under SCFGs than under STSGs (in general). On the other hand, we will also show that when the input to the SCFG becomes ambiguous (i.e., an SFSA), the selection of the most probable sentence is also NP-hard (this is related to formula (2)). The latter problem emerges, e.g., in applications of speech-understanding that involve an SCFG-based language model. This shows that the hard problems of ambiguity resolution are not so much inherent to the more advanced grammar formalisms as much as to the construction of the problem as optimization over summation formulae as in e.g., formula 5. In the next section we define each of the problems more formally.

## 4. Definition of the Optimization Problems

In this section we state the four optimization problems that this study concerns. Subsequently, each of these problems is transformed into a suitable decision problem that will be proven to be NP-complete in Section 5.

### 4.1. DISAMBIGUATION PROBLEMS

**Problem *MPP*:**

**INSTANCE:** An STSG $G$ and a sentence $w_0^n$.

**QUESTION:** What is the Most Probable Parse (MPP) of sentence $w_0^n$ under STSG $G$?, i.e., compute $\arg\max_T P_G(T, w_0^n)$.

[12] Assuming that the choice of what non-terminal (in the current sentential-form) to expand at any derivation-step is fixed, e.g., left-most.

This problem was put forward by Bod (1995b). As mentioned earlier, problem *MPP* plays a prominent role in various applications: in order to derive the semantics of an input sentence, most NLP systems (and the linguistic theories they are based on) assume that one needs a syntactic representation of that sentence. Under most linguistic theories, the syntactic representation is a parse-tree. For various probabilistic models (Bod, 1995a; Goodman, 1998; Schabes, 1992; Resnik, 1992), it can be shown that (under certain methods for estimating the model parameters from data) the MPP is the parse that maximizes the exact tree-match measure (under the given model) (Goodman, 1998).

**Problem** *MPPWG***:**

**INSTANCE:** An STSG $G$ and an $\mathcal{S}_\mathcal{I}\mathcal{FSA}$ $\mathcal{W}G$.

**QUESTION:** What is the MPP of $\mathcal{W}G$ under[13] STSG $G$?, i.e., compute $\arg\max_T P_G(T, \mathcal{W}G)$.

Applications of this problem are similar to the applications of problem *MPP*. In these applications, the input is ambiguous and the parser must select the most probable of the set of parses of all sentences which the $\mathcal{S}_\mathcal{I}\mathcal{FSA}$ generates. By selecting the MPP, the parser selects a sentence of the $\mathcal{S}_\mathcal{I}\mathcal{FSA}$ also. Typical applications lie in Speech Understanding (Oeder and Ney, 1993), morphological analysis (Sima'an et al., 2001), but also in parsing the ambiguous output of a part-of-speech (PoS) tagger[14] or morphological analyzer which provides more than one solution for every word in the input sentence.

**Problem** *MPS***:**

**INSTANCE:** An STSG $G$ and an $\mathcal{S}_\mathcal{I}\mathcal{FSA}$ $\mathcal{W}G$.

**QUESTION:** What is the Most Probable Sentence (MPS) in $\mathcal{W}G$ under STSG $G$?, i.e., compute $\arg\max_U P_G(U, \mathcal{W}G)$.

This problem has applications that are similar to problem *MPPWG*. In Speech Understanding it is often argued that the language model should select the MPS rather than the MPP of the input $\mathcal{S}_\mathcal{I}\mathcal{FSA}$. Selecting the MPS, however, does not entail the selection of a syntactic structure for the sentence (which is important for further interpretation).

**Problem** *M***PS-SCFG:**

[13] A *parse generated for an* $\mathcal{S}_\mathcal{I}\mathcal{FSA}$ *by some grammar* is a parse generated by the grammar for a sentence that is generated by the $\mathcal{S}_\mathcal{I}\mathcal{FSA}$. In formal terms we should speak of a sentence in the intersection between the language of $\mathcal{S}_\mathcal{I}\mathcal{FSA}$ and that of the grammar (van Noord, 1995). We will be slightly less formal than this to avoid complicating the notation more than necessary. Hence, we use the joint probability notation $P_G(T, \mathcal{W}G)$ with the understanding that this is a set containing all $P_G(T, U)$ for every sentence $U$ generated by $\mathcal{W}G$.

[14] PoS-taggers assign to input words their lexical categories similar to lexical-analyzers that are used in compilers for programming languages.

**INSTANCE:** An *SCFG* $G$ and an $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}$ $\mathcal{W}G$.

**QUESTION:** What is the MPS in $\mathcal{W}G$ under SCFG $G$?, i.e., compute $\arg\max_U P_G(U, \mathcal{W}G)$.

This problem is a special case of problem *MPS* and applications that face this problem are similar to those that face problem *MPS* described above. The proof that this problem is also NP-complete shows that the source of hardness of *MPS* is not inherent to the fact that the language model involves an STSG.

### 4.2. THE CORRESPONDING DECISION PROBLEMS

The decision problems that correspond to problems *MPP*, *MPPWG*, *MPS* and *M*PS-SCFG are given the same names. In these decision problems, we will supply with every problem instance a "threshold" (or "lower bound") $0 < Q \leqslant 1.0$ on the value of the optimization formula. Then, instead of asking for the maximum value, we now ask whether there is any value which exceeds $Q$.

**Decision problem *MPP*:**

**INSTANCE:** An STSG $G$, a sentence $w_0^n$ and a real number $0 < Q \leqslant 1$.

**QUESTION:** Does STSG $G$ generate for sentence $w_0^n$ a parse $T$ for which it assigns a probability $P_G(T, w_0^n) \geqslant Q$?

**Decision problem *MPPWG*:**

**INSTANCE:** An STSG $G$, an $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}$ $\mathcal{W}G$ and a real number $0 < Q \leqslant 1$.

**QUESTION:** Does STSG $G$ generate for $\mathcal{W}G$ a parse $T$ for which it assigns a probability $P_G(T, \mathcal{W}G) \geqslant Q$?

**Decision problem *MPS*:**

**INSTANCE:** An STSG $G$, an $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}$ $\mathcal{W}G$ and a real number $0 < Q \leqslant 1$.

**QUESTION:** Does $\mathcal{W}G$ accept a sentence $U$ for which STSG $G$ assigns a probability $P_G(U, \mathcal{W}G) \geqslant Q$?

**Decision problem *M*PS-SCFG:**

**INSTANCE:** An *SCFG* $G$, an $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}$ $\mathcal{W}G$ and a real number $0 < Q \leqslant 1$.

**QUESTION:** Does $\mathcal{W}G$ accept a sentence $U$ for which SCFG $G$ assigns a probability $P_G(U, \mathcal{W}G) \geqslant Q$?

## 5. NP-Completeness Proofs

In order to prove the NP-completeness of a given problem, it is sufficient to prove that the problem is NP-hard and is also a member of the class NP. For proving the NP-hardness of the problem, it is sufficient to exhibit a deterministic polynomial-time reduction from every instance of 3SAT to some instance of the problem at hand. This is what we do next for each of the decision problems listed in the preceding section. To this end, we restate the generic instance INS of 3SAT.

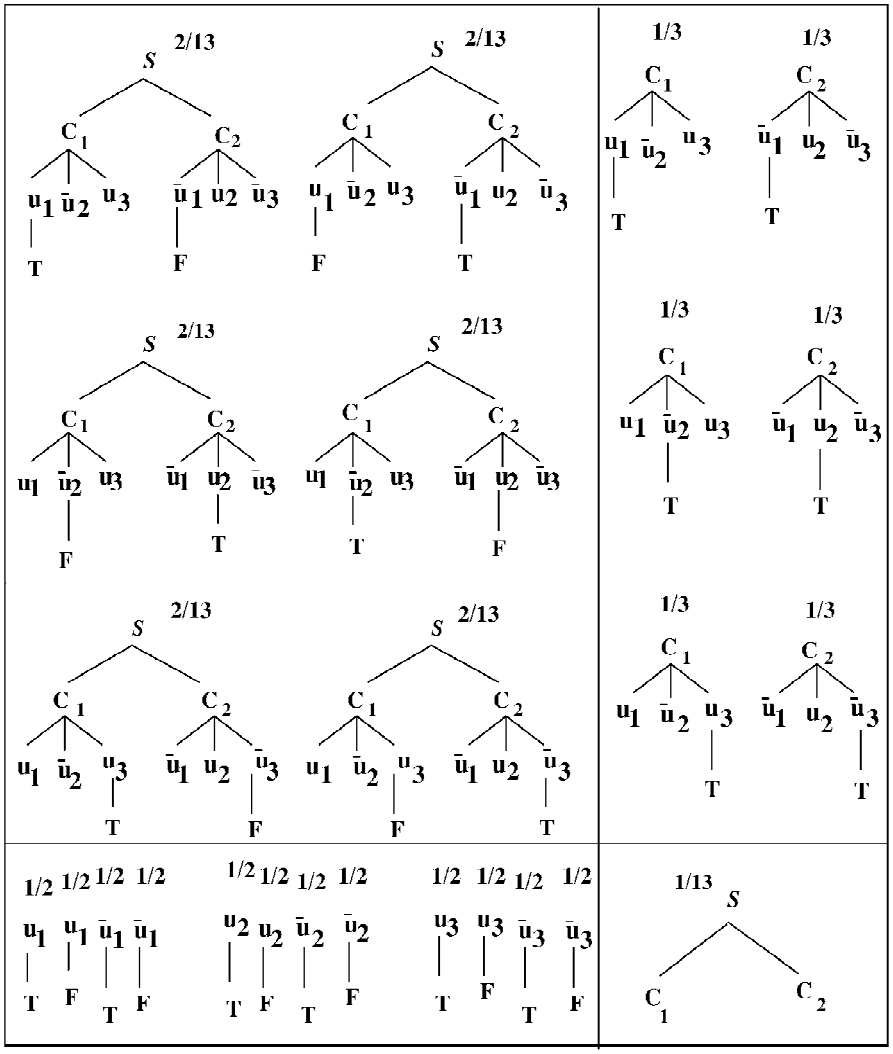


*Figure 2.* The elementary-trees for the example 3SAT instance.

**INSTANCE:** A Boolean formula in 3-conjunctive normal form (3CNF) over the variables $u_1, \ldots, u_n$:

$$(d_{11} \vee d_{12} \vee d_{13}) \wedge (d_{21} \vee d_{22} \vee d_{23}) \wedge \cdots \wedge (d_{m1} \vee d_{m2} \vee d_{m3}),$$

where $m \geqslant 1$ and $d_{ij}$ is a literal over $\{u_1, \ldots, u_n\}$, for all $1 \leqslant i \leqslant m$ and all $1 \leqslant j \leqslant 3$. This formula is also denoted $C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where $C_i$ represents $(d_{i1} \vee d_{i2} \vee d_{i3})$, for all $1 \leqslant i \leqslant m$.

**QUESTION:** Is the formula in INS *satisfiable*?

PROPOSITION 5.1. *Decision problems MPP, MPPWG, MPS and MPS-SCFG are NP-complete.*

## 5.1. A GUIDE TO THE REDUCTIONS

The reductions in the next section are structured as follows. The first two reductions are conducted simultaneously *from 3SAT to MPPWG* and *from 3SAT to MPS*. Then the reductions *from 3SAT to MPP* and *from 3SAT to MPS-SCFG* are obtained from the preceding reduction by some minor changes.

## 5.2. 3SAT TO *MPPWG* AND *MPS* SIMULTANEOUSLY

In the following, a reduction is devised which proves that both *MPPWG* and *MPS* are NP-hard. For convenience, the discussion concentrates on problem *MPPWG*, but also explains why the same reduction is suitable also for *MPS*.

The reduction from the 3SAT instance INS to an *MPPWG* instance must construct an STSG, an $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}$ and a threshold value $Q$ in deterministic polynomial-time. Moreover, the answers to the *MPPWG* instance must correspond exactly to the answers to INS. The presentation of the reduction shall be accompanied by an example of the following 3SAT instance (Barton et al., 1987):

$$(u_1 \vee \overline{u}_2 \vee u_3) \wedge (\overline{u}_1 \vee u_2 \vee \overline{u}_3),$$

where $u_1$, $u_2$ and $u_3$ are Boolean variables.

### 5.2.1. *Some Intuition*

A 3SAT instance is satisfiable iff at least one of the literals in each conjunct is assigned the value *true*. Implicit in this, but crucial, the different occurrences of the literals of the same variable must be assigned values *consistently*. These two observations constitute the basis of the reduction. The reduction must capture these two "satisfiability-requirements" of INS in the problem-instances that it constructs. For example, for *MPPWG* we will construct an STSG and an $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}$. The $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}$ will be $\mathcal{WG} = \{\mathbf{T}, \mathbf{F}\}_1^{3m}$, where $3m$ is the number of literals in the formula of INS. The STSG will be constructed such that it has two kinds of derivations for every path (in $\mathcal{WG}$) which constitutes a solution for INS (if INS is satisfiable): one kind of derivations takes care of the consistent assignment of truth values, and the other takes care of the assignment of the value *true* for exactly one literal in every conjunct. Crucially, the derivations will be assigned such probabilities that will enable us to know whether a path in $\mathcal{WG}$ is a solution for INS by inspecting the probability of that path. In other words, the probability of a path in $\mathcal{WG}$ will tell us whether the STSG derives that path by *enough derivations of each of the two kinds of derivations* in order for that path to be a solution for INS.

### 5.2.2. *The Reduction*

The reduction constructs an STSG and an $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}$. The STSG has start-symbol labeled $\mathcal{S}$, two terminals represented by **T** and **F**, non-terminals which include (beside $\mathcal{S}$) all $C_k$, for $1 \leqslant k \leqslant m$, and both literals of each Boolean variable of the formula of INS. The set of elementary-trees and probability function and the $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}$ are constructed as follows:

1. The reduction constructs for each Boolean variable $u_i$, $1 \leqslant i \leqslant n$, two elementary-trees that correspond to assigning the values *true* (**T**) and *false* (**F**) to $u_i$ *consistently* through the whole formula. Each of these elementary-trees has root $\mathcal{S}$, with children $C_k$, $1 \leqslant k \leqslant m$, in the same order as they appear in the formula of INS; subsequently the children of $C_k$ are the non-terminals that correspond to its three disjuncts $d_{k1}$, $d_{k2}$ and $d_{k3}$. And, finally, the assignment of *true* (*false*) to $u_i$ is achieved by creating a child terminal **T** (resp. **F**) to each non-terminal $u_i$ and **F** (resp. **T**) to each $\overline{u}_i$. The two elementary-trees for $u_1$, of our running example, are shown in the top left corner of figure 2.
2. The reduction constructs three elementary-trees for each conjunct $C_k$. The three elementary-trees for conjunct $C_k$ have the same internal structure: root $C_k$, with three children that correspond to the disjuncts $d_{k1}$, $d_{k2}$ and $d_{k3}$. In each of these elementary-trees exactly one of the disjuncts has a child labeled with the terminal **T**; in each of the three elementary-trees this **T** child is a different one. Each of these elementary-trees corresponds to the given conjunct where one of the three possible literals is assigned the value **T**. For the elementary-trees which are constructed for our example see the top right corner of figure 2.
3. The reduction constructs for each of the literals of each variable $u_i$ two elementary-trees where the literal is assigned in one case **T** and in the other **F**. Figure 2 shows these elementary-trees for variable $u_1$ in the bottom left corner.
4. The reduction constructs one elementary-tree that has root $\mathcal{S}$ with children $C_k$, $1 \leqslant k \leqslant m$, in the same order as these appear in the formula of INS (see the bottom right corner of Figure 2).
5. The reduction assigns probabilities to the elementary-trees that were constructed by the preceding steps. The probabilities of the elementary-trees that have the same root non-terminal must sum up to 1. The probability of an elementary-tree with root label $\mathcal{S}$ that was constructed in step 1 of this reduction is a value $p_i$, $1 \leqslant i \leqslant n$, where $u_i$ is the only variable of which the literals in the elementary-tree at hand are lexicalized (i.e., have terminal children). Let $n_i$ denote the number of occurrences of both literals of variable $u_i$ in the formula of INS. Then $p_i = \theta \left(\frac{1}{2}\right)^{n_i}$, for some real $\theta$ that has to fulfill some conditions which will be derived in Section 5.2.3.

   The probability of the tree rooted with $\mathcal{S}$ and constructed at step 4 of this reduction must then be $p_0 = [1 - 2\sum_{i=1}^{n} p_i]$.

The probability of the elementary-trees of root $C_k$ (step 2) is $\left(\frac{1}{3}\right)$, and of root $u_i$ or $\overline{u}_i$ (step 3) is $\left(\frac{1}{2}\right)$. Suitable probabilities are shown in figure 2 for our running example.

Let $Q$ denote a threshold probability that shall be derived below. The *MPPWG* (*MPS*) instance produced by this reduction is:

**INSTANCE:** The STSG produced by the above reduction (probabilities are derived below), the $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}$ $\mathcal{WG} = \{\mathbf{T}, \mathbf{F}\}_1^{3m}$, and a probability threshold $0 < Q \leqslant 1$.

**QUESTION:** Does this STSG generate for the $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}$ $\mathcal{WG} = \{\mathbf{T}, \mathbf{F}\}_1^{3m}$ a parse (resp. a sentence) for which it assigns a probability greater than or equal to $Q$?

### 5.2.3. *Deriving the Probabilities and the Threshold*

The parses generated by the constructed STSG differ only in the sentences on their frontiers. Therefore, if a sentence is generated by this STSG then *it has exactly one parse*. This clarifies why we choose to reduce 3SAT to *MPPWG* and *MPS* simultaneously.

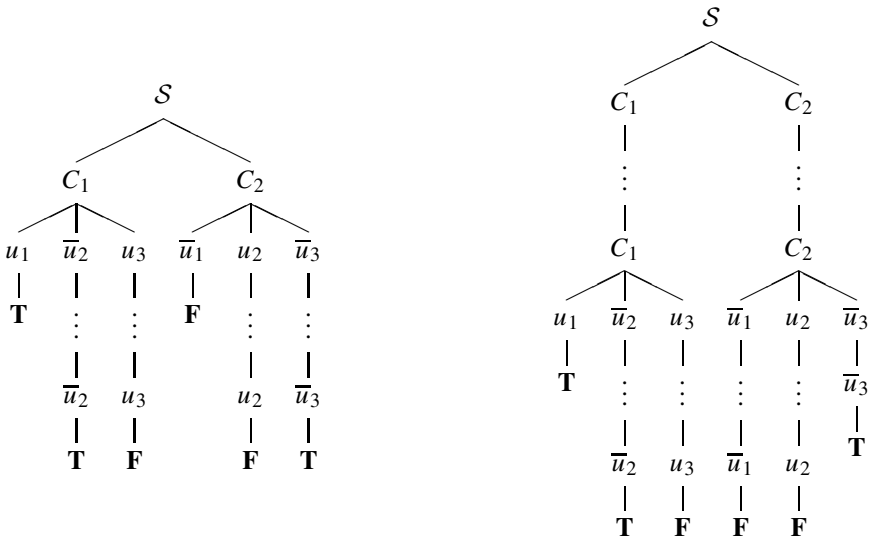


*Figure 3.* Left: example of the first kind of derivations. Right: example of the second kind. In this figure, the vertical dots signify substitutions.

By inspecting the STSG resulting from the reduction, one can recognize exactly two types of derivations in this STSG (see Figure 3):

1. The first type corresponds to substituting a suitable elementary-tree for a non-terminal leaf node (i.e., literal) in any of the $2n$ elementary-trees constructed in step 1 of the reduction. This type of derivation corresponds to the *consistent*

*assignment* of values to all literals of some variable $u_i$. For all $1 \leqslant i \leqslant n$ the probability of a derivation of this type is:

$$p_i \left(\frac{1}{2}\right)^{3m-n_i} = \theta \left(\frac{1}{2}\right)^{3m}$$

2. The second type of derivation corresponds to substituting the elementary-trees that has $C_k$ as root for the leaf-node labeled $C_k$ in $\mathcal{S} \rightarrow C_1 \ldots C_m$, and subsequently substituting suitable elementary-trees for the non-terminal leaf-nodes that correspond to literals. This type of derivation corresponds to the assignment of the value *true* to at least one literal in each conjunct. The probability of any such derivation is:

$$p_0 \left(\frac{1}{2}\right)^{2m} \left(\frac{1}{3}\right)^{m} = [1 - 2\theta \sum_{i=1}^{n} \left(\frac{1}{2}\right)^{n_i}] \left(\frac{1}{2}\right)^{2m} \left(\frac{1}{3}\right)^{m}$$

Next we derive both the threshold $Q$ and the parameter $\theta$. Any parse (or sentence) that fulfills both the "consistency of assignment" requirements and the requirement that each conjunct has at least one literal with child **T**, must be generated by $n$ derivations of the first type and at least one derivation of the second type. Note that a parse can never be generated by more than $n$ derivations of the first type. Thus the threshold $Q$ must be set at:

$$Q = n\theta \left(\frac{1}{2}\right)^{3m} + [1 - 2\theta \sum_{i=1}^{n} \left(\frac{1}{2}\right)^{n_i}] \left(\frac{1}{2}\right)^{2m} \left(\frac{1}{3}\right)^{m}$$

However, $\theta$ must fulfill some requirements for our reduction to be acceptable:

1. For all i: $0 < p_i < 1$. Hence, for $1 \leqslant i \leqslant n$: $0 < \theta \left(\frac{1}{2}\right)^{n_i} < 1$, and $0 < p_0 < 1$. However, the last requirement on $p_0$ implies that $0 < 2\theta \sum_{i=1}^{n} \left(\frac{1}{2}\right)^{n_i} < 1$, which is a stronger requirement than the other $n$ requirements. This requirement can also be stated as follows:

$$0 < \theta < \frac{1}{2 \sum_{i=1}^{n} \left(\frac{1}{2}\right)^{n_i}}$$

2. Since we want to be able to know whether a parse is generated by a second type derivation only by looking at the probability of the parse, the probability of a second type derivation must be distinguishable from first type derivations. Moreover, if a parse is generated by more than one derivation of the second type, we do not want the sum of the probabilities of these derivations to be mistaken for one (or more) first type derivation(s). For any parse, there are at most $3^m$ second type derivations (e.g., the sentence **T**...**T**). Therefore we

require that: $3^m[1 - 2\theta \sum_{i=1}^{n} \left(\frac{1}{2}\right)^{n_i}] \left(\frac{1}{2}\right)^{2m} \left(\frac{1}{3}\right)^{m} < \theta \left(\frac{1}{2}\right)^{3m}$. Hence:

$$\theta > \frac{1}{2\sum_{i=1}^{n} \left(\frac{1}{2}\right)^{n_i} + \left(\frac{1}{2}\right)^{m}}$$

3. For the resulting STSG to be a probabilistic model, the "probabilities" of parses and sentences must be in the interval (0, 1]. This is taken care of by demanding that the sum of the probabilities of elementary-trees that have the same root non-terminal is 1, and by the definition of the derivation's probability, the parse's probability, and the sentence's probability.

There exists a $\theta$ that fulfills the requirements expressed in the inequalities 1 and 2 because the lower bound $1/2 \sum_{i=1}^{n} \left(\frac{1}{2}\right)^{n_i} + \left(\frac{1}{2}\right)^{m}$ is always larger than zero and is *strictly smaller* than the upper bound $1/2 \sum_{i=1}^{n} \left(\frac{1}{2}\right)^{n_i}$.

5.2.3.1. *Polynomiality of the reduction:* This reduction is deterministic polynomial-time in $m$ and $n$ (note that $n \leqslant 3m$ always). It constructs not more than $2n + 1 + 3m + 4n$ elementary-trees, each consisting of at most $7m + 1$ nodes. And the computation of the probabilities and the threshold is also conducted in deterministic polynomial-time. Furthermore, the construction of the $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}\{\mathbf{T}, \mathbf{F}\}_1^{3m}$ is polynomial[15] in $m$.

5.2.3.2. *The reduction preserves answers:* The proof that this reduction preserves answers concerns the two possible answers Yes and No:

Yes: If INS's answer is Yes then there is an assignment to the variables that is consistent and where each conjunct has at least one literal assigned *true*. Any possible assignment is represented by one sentence in $\mathcal{WG}$. A sentence which corresponds to a "successful" assignment must be generated by $n$ derivations of the first type and at least one derivation of the second type; this is because the sentence $w_1^{3m}$ fulfills $n$ consistency requirements (one per Boolean variable) and has at least one **T** assignment for any of $w_{3k+1}$, $w_{3k+2}$ or $w_{3k+3}$, for all $0 \leqslant k < m$. Both this sentence and its corresponding parse have probabilities $\geqslant Q$. Thus *MPPWG* and *MPS* also answer Yes.

No: If INS's answer is No, then all possible assignments are either not consistent or result in at least one conjunct with three false disjuncts, or both. The sentences (parses) that correspond to non-consistent assignments each have a probability that cannot result in a Yes answer. This is the case because such sentences have fewer than $n$ derivations of the first type, and the derivations of the second type can never compensate for that (the requirements on $\theta$ take care of this). For the sentences (parses) that correspond to consistent assignments, there is at least some $0 \leqslant k < m$ such that $w_{3k+1}$, $w_{3k+2}$ and $w_{3k+3}$ are all **F**. These sentences

[15] Recall that the notation $Q_1^m$ is a shorthand for $Q_1, \cdots, Q_m$.

do not have second type derivations. Thus, there is no sentence (parse) that has a probability that can result in a Yes answer; the answer of *MPPWG* and *MPS* is NO. □

To summarize, we have deterministic polynomial-time reductions, that preserve answers, from 3SAT to *MPPWG* and from 3SAT to *MPS*. We conclude that *MPPWG* and *MPS* are both NP-hard problems. Next we prove NP-completeness in order to show that these problems are *deterministic polynomial-time solvable iff P = NP*.

#### 5.2.4. *NP-completeness of MPS and MPPWG*

We show that *MPPWG* and *MPS* are in NP. A problem is in NP if it is decidable by a non-deterministic Turing Machine in polynomial-time. In general, however, it is possible to be less formal than this. It is sufficient to exhibit a suitable non-deterministic algorithm, which does the following: it proposes some entity as a solution (e.g., a parse for *MPPWG* and a sentence for *MPS*), and then computes an answer (Yes/NO) to the question of the decision problem, on this entity, in deterministic polynomial-time (cf. Garey and Johnson, 1981; Barton et al., 1987). The non-deterministic part lies in guessing or proposing an entity as a solution.

For *MPPWG*, a possible algorithm proposes a parse, from the set of all parses which the STSG $G$ assigns to $\mathcal{W}G$, and computes its probability in deterministic polynomial-time (Sima'an, 1999) and verifies whether this probability is larger or equal to the threshold $Q$. Similarly, an algorithm for *MPS* proposes a sentence, from those accepted by the $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}$, and computes its probability in polynomial-time (Sima'an, 1999) and then verifies whether this probability is larger or equal to the threshold $Q$. In total, both algorithms are non-deterministic polynomial-time, which proves that *MPPWG* and *MPS* are both in class NP.

In summary, now we proved that decision problems *MPPWG* and *MPS* are both NP-hard and in NP, thus both are NP-complete. Therefore, the corresponding optimization problems *MPPWG* and *MPS* are NP-hard.

### 5.3. NP-COMPLETENESS OF *MPP*

The NP-completeness of *MPP* can be easily deduced from the proof in Section 5.2. The proof of NP-hardness of *MPP* is based on a reduction from 3SAT to *MPP* obtained from the preceding reduction by minor changes. The main idea now is to construct a sentence and a threshold, and to adapt the STSG in such a way that the sentence has many parses, each corresponding to some possible assignment of truth values to the literals of the 3SAT instance. As in the preceding reduction, the STSG will have at most two kinds of derivations and suitable probabilities; again the probabilities of the two kinds of derivations enable inspecting whether a parse is generated by enough derivations that it corresponds to an assignment that

satisfies the 3SAT instance, i.e., a consistent assignment that assigns the value *true* to at least one literal in every conjunct.

The preceding reduction is adapted as follows. In the present reduction, the terminals of the constructed STSG are fresh new symbols $v_{ij}$, $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant 3$; the symbols **T** and **F** are now non-terminals of the STSG (instead of terminals as in the preceding reduction). Then, some of the elementary-trees and some of the probabilities, constructed by the preceding reduction, are adapted as follows (any entity not mentioned below remains exactly the same):

1. In each elementary-tree, constructed in step 1 or step 2 of the preceding reduction (with root node labeled either $\mathcal{S}$ or $C_k$), the leaf node labeled **T/F** (a non-terminal), which is the child of the $j$th child (a literal) of $C_k$, now has a child node labeled $v_{kj}$.
2. Instead of each of the elementary-trees that have a root labeled with a literal (i.e., $u_k$ or $\overline{u}_k$), which were constructed in step 3 of the preceding reduction, there are now $3m$ elementary-trees, each corresponding to adding a terminal-child $v_{ij}$, $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant 3$, under the node labeled **T/F** (previously a leaf node).
3. The probability of an elementary-tree rooted by a literal (adapted in the preceding step) is now $1/6m$. The probabilities of elementary-trees rooted with $C_k$ do not change. The probabilities of the elementary-trees that have a root labeled $\mathcal{S}$ are adapted from the previous reduction by substituting for every $(1/2)$ the value $1/6m$.
4. The threshold $Q$ and the requirements on $\theta$ are updated accordingly, and then derived as done for the preceding reduction.
5. The decision problem:

   **INSTANCE:** The STSG constructed in this reduction, sentence $v_{11} \ldots v_{m3}$ and a probability threshold $0 < Q \leqslant 1$.

   **QUESTION:** Does this STSG generate for sentence $v_{11} \ldots v_{m3}$ any parse-tree that has probability larger than or equal to $Q$?

The proofs that this reduction is polynomial-time and answer-preserving are very similar to that in Section 5.2. It is easy also to prove that *MPP* is in class NP (very much like *MPPWG*). Therefore, the decision problem *MPP* is NP-complete and the corresponding optimization problem is NP-hard.

### 5.4. NP-COMPLETENESS OF *M*PS-SCFG

The decision problem *MPS* is NP-complete also under SCFG, i.e., *M*PS-SCFG is NP-complete. The proof is easily deducible from the proof concerning *MPS* for STSGs by a simple adaptation of the reduction for *MPS*. Every elementary-tree of the *MPS* reduction is now simplified by "masking" its internal structure, thereby obtaining simple CFG rules/productions, i.e., elementary-trees of one level.

Crucially, each elementary-tree results in one unique CFG production. The probabilities are kept the same and also the threshold. The $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}$ is also the same $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}$ as in the reduction of *MPS*. The present decision-problem's question is: *does the thus created SCFG generate a sentence with probability* $\geqslant Q$*, accepted by the* $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}$ $\mathcal{WG} = \{\mathbf{T}, \mathbf{F}\}_1^{3m}$.

Note that for each derivation, which is possible in the STSG of problem *MPS* there is one corresponding unique derivation in the SCFG of problem *M*PS-SCFG. Moreover, there are no extra derivations. Of course, each derivation in the SCFG of problem *M*PS-SCFG generates a different parse. But that does not affect the probability of a sentence at all: it remains the sum of the probabilities of all derivations that generate it in the SCFG. The rest of the proof follows directly from Section 5.2. Therefore, computing the *MPS* of an $\mathcal{S}_{\mathcal{I}}\mathcal{FSA}$ for SCFGs is also NP-complete and the corresponding optimization problem is NP-hard.

## 6. Discussion and Conclusions

We provided proofs that probabilistic disambiguation problems *MPP*, *MPPWG* and *MPS* are NP-hard. As a direct result of the proofs, we conclude that probabilistic disambiguation of language utterances, as specified in Section 4, under models based on STSGs and STAGs is (as far as we know) not solvable by deterministic polynomial-time algorithms. Examples of models based on STSGs are Bod (1992, 1995a), Sekine and Grishman (1995), Bod and Kaplan (1998), Sima'an (2000) and on STAGs are Schabes (1992), Resnik (1992), Schabes and Waters (1993), Chiang (2000). Maybe more surprising is the fact that we proved that problem *M*PS-SCFG is NP-complete. Problem *M*PS-SCFG applies SCFGs, i.e., the "shallowest" of all STSGs, for the disambiguation of "ambiguous" inputs, in the form of Finite State Automata (FSAs), e.g., word-graphs output by speech-recognizers Oeder and Ney (1993). Example models that (formally) employ SCFGs are Jelinek et al. (1990), Black et al. (1993), Charniak (1996, 1999), Collins (1996, 1997), Ratnaparkhi (1997).

The proofs also provide some insight into why the problems are so hard to solve efficiently. The fact that computing the *MPS* of an FSA under SCFGs is also NP-complete implies that the complexity of these problems is not due to the kind of grammar underlying the models. Rather, the main source of NP-completeness is the following common structure for these problems:

> The probability of the entity which the problem aims at optimizing is expressed in terms of the sum of the probabilities of multiple (possibly exponentially many) stochastic processes that generate that entity.

For example, in problem *M*PS-SCFG, the model searches for the sentence, in a finite, yet possibly exponential, set accepted by an FSA, which maximizes the *sum* of the probabilities of *the derivations that generate that sentence*.

There are some practical situations where exponential algorithms can be useful. This is usually the case if the exponential function exhibits growth that is similar to

that of a "low degree" polynomial. Unfortunately, however, for parsing algorithms for natural language grammars, the common exponentials, concerning derived entities, e.g., the number of possible parse-trees, exhibit a growth function which exceeds by far the growth of so called "low degree" polynomials. In Martin et al. (1987), the authors conduct an empirical study on two language corpora and conclude that the number of parses for a sentence, under a realistic linguistic grammar, can be described by the *Catalan series* and in some cases by the *Fibonacci series*. The authors conclude:

> $\cdots$, many of these series grow quickly; it may be impossible to enumerate these numbers beyond the first few values. $\cdots$

For applications such as speech-understanding, the situation is at least as bad! Typical "word-graphs" output by a speech-recognizer (for a small lexicon of 2000 words) complicate both input-length and the ambiguity factor by multiplying the complexity function at least by a large constant – which can be exponential in the length of the actually spoken-utterance.

It is unclear what conclusions can be drawn from NP-hardness results with respect to the "goodness" of the probabilistic component in existing language models. In any case, it is clear that having proved the NP-completeness of these problems does not imply that we should avoid them. Rather, we think that the proofs imply that for practical situations, it might be necessary to resort to non-standard and non-conventional solutions. For example, (Bod, 1993, 1995a) proposes Monte-Carlo sampling methods for obtaining an approximate solution for problem *MPP* under an STSG-based model. Although the proposed algorithm remains exponential-time (Goodman, 1998) in the worst-case, there is some hope that the kind of empirical distributions exhibited by linguistic data might result in observable-times that are better than exponential-time parsing. This has not been confirmed by empirical experiments, though.

Another traditional possibility is to model the extra-syntactic aspects of language processing, e.g., semantic interpretation, discourse structure and world-knowledge. Indeed, this could reduce the ambiguity problem in various cases. However, we suspect that this will not solve the problem completely for two reasons. Firstly, the extra-linguistic factors, such as world-knowledge, seem to be beyond complete modeling, especially because of their dynamic nature. And secondly, in applications where the input itself is ambiguous, e.g., speech understanding, such knowledge will be of little help. It seems that approximation of "what input to expect next" will remain a useful strategy for real applications. The question, however, is whether there is a way to exploit the statistics for efficient processing.

One effective way out, which has been extensively explored is the use of pruning techniques (e.g., Goodman, 1998; Caraballo and Charniak, 1998). A disadvantage of pruning, however, is the need to partially generate and evaluate various paths before one can prune them. An alternative approach is the machine learning method of "model specialization" to limited domains. In this respect, Samuelsson (1994), Sima'an (1999), Cancedda and Samuelsson (2000) suggest that probabilistic mod-

els must also capture important efficiency properties of the human language processing system in limited domains of language-use, through learning methods such as the Minimum-Description Length Principle (Rissanen, 1983; Li and Vitányi, 1997). For example, an interesting property that could be captured by a language model is to process *more expected inputs* (i.e., expected to be more frequent) more efficiently. The more "regular" the input in a given domain of language-use is, the less time and space it should demand. This kind of reasoning has been explored within small applications, leading to improved processing times, where indeed, more probable inputs are processed more efficiently. The modeling of this kind of *distributional properties* of limited language domains through precompilation techniques could prove to be a fruitful path for the sake of more efficient models of language processing.

## Acknowledgements

I thank the following people for comments on preliminary versions of this work: Remko Scha, Christer Samuelsson, Remko Bonnema, Rens Bod and Gabriele Musillo. I also thank the anonymous reviewers for their comments. This work was conducted within a project of the Netherlands Organization for Scientific Research (NWO) and a following project of the Royal Dutch Academy for Arts and Sciences (KNAW).

## References

Barton, G. E., R. Berwick and E. S. Ristad. *Computational Complexity and Natural Language*, A Bradford Book, MIT Press, Cambridge, MA, 1987.

Black, E., F. Jelinek, J. Lafferty, D. Magerman, R. Mercer and S. Roukos. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings ACL'93*, Columbus, Ohio, 1993.

Bod, R. A computational model of language performance: Data Oriented Parsing. In *Proceedings COLING'92*, Nantes, 1992.

Bod, R. Monte Carlo Parsing. In *Proceedings Third International Workshop on Parsing Technologies*, Tilburg/Durbuy, 1993.

Bod, R. *Enriching Linguistics with Statistics: Performance models of Natural Language*. PhD thesis, ILLC-Dissertation Series 1995-14, University of Amsterdam, 1995a.

Bod, R. The problem of computing the most probable tree in data-oriented parsing and stochastic tree grammars. In *Proceedings Seventh Conference of The European Chapter of the ACL*, Dublin, 1995b.

Bod, R. and R. Kaplan. A probabilistic corpus-driven approach for Lexical Functional Grammar. In *Proceedings COLING-ACL'98*, Montréal, Canada, 1998.

Cancedda, N. and C. Samuelsson. Experiments with corpus-based LFG specialization. In *Proceedings Sixth Applied Natural Language Processing Conference (ANLP 2000)*, Seattle, Washington, 2000.

Caraballo, S. and E. Charniak. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24-2: 275–298, 1998.

Charniak, E. Tree-bank Grammars. In *Proceedings AAAI'96*, Portland, Oregon, 1996.

Charniak, E. A maximum-entropy-inspired parser. In *Report CS-99-12*, Providence, Rhode Island, 1999.

Chiang, D. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings* $38^{th}$ *Annual Meeting of the Association for Computational Linguistics (ACL'00)*, 456–463, Hong Kong, China, 2000.

Chomsky, N. *Aspects of the Theory of Syntax*. MIT Press, Cambridge Massachusetts, 1965.

Collins, M. A new statistical parser based on bigram lexical dependencies. In *Proceedings 34th Annual Meeting of the ACL*, 184–191, 1996.

Collins, M. Three generative, lexicalized models for statistical parsing. In *Proceedings 35th Annual Meeting of the ACL and 8th Conference EACL*, 16–23, Madrid, Spain, 1997.

Davis, M. and E. Weyuker. *Computability, Complexity and Languages: Fundamentals of Theoretical Computer Science*. Series in Computer Science and Applied Mathematics, Academic Press, New York, 1983.

Garey, M. and D. Johnson, *Computers and Intractability*. W.H. Freeman and Co, San Francisco, 1981.

Goodman, J. *Parsing Inside-Out*. PhD thesis, Department of Computer Science, Harvard University, Cambridge, Massachusetts, 1998.

Hopcroft, J. and J. Ullman. *Introduction to Automata Theory, Lanaguges, and Computation*, Addison Wesley, Reading, Massachusetts, 1979.

Jelinek, F., J. Lafferty, D. Magerman, R. Mercer, A. Ratnaparkhi and S. Roukos. Decision tree parsing using a hidden derivation model. In *Proceedings 1994 Human Language Technology Workshop*. DARPA, 1994.

Jelinek, F., J. Lafferty and R. Mercer. *Basic Methods of Probabilistic Context Free Grammars*, Technical Report IBM RC 16374 (#72684). Yorktown Heights, 1990.

Joshi, A. Tree adjoining grammars: How much context sensitivity is required to provide a reasonable structural description. In D. Dowty, L. Karttunen and A. Zwicky, editors, *Natural Language Parsing*, 206–250, Cambridge University Press, Cambridge, 1985.

Lewis, H. and C. Papadimitriou. *Elements of the Theory of Computation*, Prentice-Hall, Englewood-Cliffs, NJ, 1981.

Li, M. and P. Vitányi. *An Introduction to Kolomogorov Complexity and Its Applications*, 2nd edition, Springer, Berlin, 1997.

Magerman, D.M. Statistical decision-tree models for parsing. In *Proceedings 33*rd *Annual Meeting of the ACL*, 1995.

Marcus, M., B. Santorini and M. Marcinkiewicz. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19: 313–330, 1993.

Martin, W., K. Church and R. Patil. Preliminary analysis of a Breadth-First Parsing Algorithm: Theoretical and Experimental Results. In Bolc, L., editor, *Natural Language Parsing Systems*, 267–328, Springer, Berlin, 1987.

Oeder, M. and H. Ney. Word graphs: An efficient interface between continuous-speech recognition and language understanding. In *ICASSP*, volume 2, 119–122, 1993.

Pereira, F. and Y. Schabes. Inside-outside reestimation from partially bracketed corpora. In *Proceedings 30th Annual Meeting of the ACL*, Newark, Delaware, 1992.

Rabiner, L.R. and B.H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*: 4–15, 1986.

Ratnaparkhi, A. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of Empirical Methods in NLP, EMNLP-2*, 1–10, 1997.

Resnik, P. Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proceedings COLING'92*, Nantes, 1992.

Rissanen, J. A universal prior for integers and estimation by minimum description length, *The Annuals of Statistics*, 11 (2): 416–431, 1983.

Salomaa, A. Probabilistic and weighted grammars, *Information and Control*, 15: 529–544, 1969.

Samuelsson, C. *Fast Natural-Language Parsing Using Explanation-Based Learning*, Swedish Institute of Computer Science Dissertation Series 13, Stockholm, Sweden, 1994.

Scha, R. Language theory and language technology; competence and performance. In de Kort, Q. and Leerdam, G., editors, *Computertoepassingen in de Neerlandistiek*, Almere: LVVN-jaarboek (can be obtained from http://www.hum.uva.nl/computerlinguistiek /scha/IAAA/rs/cv.html#Linguistics), 1990.

Schabes, Y. Stochastic lexicalized tree-adjoining grammars. In *Proceedings COLING'92*, Nantes, 1992.

Schabes, Y. and R. Waters. Stochastic lexicalized context-free grammar. In *Proceedings Third IWPT*, Tilburg/Durbuy, 1993.

Sekine, S. and Grishman, R. A corpus-based probabilistic grammar with only two non-terminals. In *Proceedings Fourth International Workshop on Parsing Technologies*, Prague, 1995.

Sima'an, K. *Learning Efficient Disambiguation*. PhD dissertation. ILLC Dissertation Series 1999-02 (Utrecht University / University of Amsterdam), Amsterdam, 1999.

Sima'an, K. Tree-gram parsing: lexical dependencies and structural relations. In *Proceedings* $38^{th}$ *Annual Meeting of the Association for Computational Linguistics (ACL'00)*, 53–60, Hong Kong, China, 2000.

Sima'an, K., A. Itai, Y. Winter, A. Altman and N. Nativ. Building a tree-bank of modern Hebrew texts. *Traitement Automatique des Langues*, Special Issue on Natural Language Processing and Corpus Linguistics (Béatrice Daille and Laurent Romary eds.), 42(2), 2001.

van Noord, G. The intersection of finite state automata and definite clause grammars. In *Proceedings ACL-95*, 1995.