Bericht Nr. 117, November 1997

Arbeitspapiere des
Sonderforschungsbereichs 340

ISSN 0947-6954/97

Sprachtheoretische Grundlagen für die Computerlinguistik

Stefan Riezler :

Probabilistic Constraint Logic Programming

Universität Stuttgart   Universität Tübingen
IBM Deutschland Informationssysteme GmbH

Arbeitspapiere des SFB 340
Bericht Nr. 117, Oktober 1997

PROBABILISTIC CONSTRAINT LOGIC PROGRAMMING

STEFAN RIEZLER

ABSTRACT. This paper addresses two central problems for probabilistic processing models: parameter estimation from incomplete data and efficient retrieval of most probable analyses. These questions have been answered satisfactorily only for probabilistic regular and context-free models. We address these problems for a more expressive probabilistic constraint logic programming model.

We present a log-linear probability model for probabilistic constraint logic programming. On top of this model we define an algorithm to estimate the parameters and to select the properties of log-linear models from incomplete data. This algorithm is an extension of the improved iterative scaling algorithm of Della Pietra, Della Pietra, and Lafferty (1995). Our algorithm applies to log-linear models in general and is accompanied with suitable approximation methods when applied to large data spaces. Furthermore, we present an approach for searching for most probable analyses of the probabilistic constraint logic programming model. This method can be applied to the ambiguity resolution problem in natural language processing applications.

1. INTRODUCTION

Rabiner (1989) identified three basic problems of interest that must be solved for a Hidden Markov Model to be useful in real-world speech recognition applications: the parameter estimation problem, the optimal state sequence problem and the observation sequence probability problem. These problems generalize to arbitrary probabilistic symbol processing models in various real-world applications in an obvious manner. The first two problems can be stated in a more general way as follows.

This work was supported by the Graduiertenkolleg Integriertes Linguistikstudium at the University of Tübingen and by the Teilprojekt B7 of the Sonderforschungsbereich 340 of the Deutsche Forschungsgemeinschaft.

I am greatly indebted to Steven Abney for coaching this work in many helpful discussions. Furthermore, I would like to thank Graham Katz and Detlef Prescher for their valuable comments.

ISSN 0947-6954/97

1. Let an unanalyzed observation sequence $O = O_1, \ldots O_n$ and a probabilistic processing model with parameter set λ be given, and suppose that the value of λ is unknown and O forms a random sample from the distribution involving λ, how can the model parameters λ be estimated?
2. Given O_i and λ, how can the most probable analysis of the input O_i be found efficiently?

Recent interest in probabilistic models of natural language processing can be attributed to the fact that solutions to the above-mentioned general problems can lead quite directly to effective, but conceptually simple and mathematically clear solutions to various problems in natural language processing.

This connection can be illustrated with the problem of ambiguity resolution (or disambiguation or parse ranking) as follows: Grammars describing a nontrivial fragment of natural language may attach a large number of different analyses to sentences of reasonable length. Since not all of these analyses are in accord with human perceptions, there is clearly a need to distinguish more plausible analyses of an input from less plausible or totally spurious ones. The simple but effective idea adopted in probabilistic grammars is to connect the plausibility of an analysis with its probability. In this vein the correct, i.e., most plausible analysis of a string is assumed to be the most probable analysis of the string. A solution to problem 1 will adapt the model parameters λ to the input corpus O and thus justify the assumption that the correct parse of a string O_i is the most probable parse of O_i as produced by the grammar parametrized by λ. A solution to problem 2 will yield an algorithm to search for the most probable parse of a given input string O_i as produced by a probabilistic grammar with parameter set λ.

Most popular approaches to solving these problems in the area of natural language processing are based on Baum's maximization technique, which is known as the "Baum-Welch algorithm" (Baum and Eagon 1967; Baum, Petrie, Soules, and Weiss 1970; Baum 1972). This algorithm estimates the parameters of a Hidden Markov Model, i.e., a stochastic regular grammar, in a framework of maximum likelihood estimation from incomplete data. This means, the parameters are iteratively reestimated until convergence to a set of values which locally maximize the likelihood function, i.e., the probability that the model assigns to the given unanalyzed observation sequence. In this sense the model parameters are adjusted to best describe a given observation sequence. The estimation algorithm can be defined inductively forwards and backwards, yielding the efficient "forward-backward algorithm"[1]. Baker (1979) generalized this algorithm to the so-called "inside-outside algorithm"[2], which efficiently estimates the parameters of a stochastic context-free grammars. Both algorithms are special instances of the EM-algorithm for maximum-likelihood estimation from incomplete data (Dempster, Laird, and Rubin 1977). A dynamic-programming approach similar to the one used in the efficient versions of the parameter estimation algorithms can be used to find the most probable analysis of stochastic context-free and stochastic regular grammars and is known as the "Viterbi-algorithm" (Viterbi 1967).

The class of algorithms based upon Baum's maximization technique includes not only regular and context-free versions but recently has been extended to, e.g., stochastic context-free grammars with bracketing constraints (Pereira and Schabes 1992) and feature-based constraints (Briscoe and Waegner 1992), stochastic depencendy grammars (Carroll and Charniak 1992) and stochastic lexicalised tree-adjoining grammars (Resnik 1992; Schabes 1992). Despite the generality of the algorithm, there are clear restrictions on the expressivity of the probabilistic processing models the algorithm can be applied to. Even if the structural operations of the probabilistic processing model may be sensitive to contextual features, this context-sensitivity has to be internal to the structural elements combined. The combination process itself has to be context-free, i.e., in terms of probability theory, different stochastic derivation choices at the same time-step have to be independent of the history of the derivation process and also independent of one another.

This fact poses a problem for attempts to build stochastic versions of grammars which are more expressive than context-free. The grammars we are interested in here are constraint logic grammars (CLGs), i.e., highly expressive constraint-based grammars formalized in a (Turing-)powerful framework of constraint logic programming (CLP)[3]. This treatment of CLGs as special applications of CLP will allow us to refer in the following to the general framework of CLP. Stochastic versions of CLP exhibit

---

[1] See Rabiner (1989) for a tutorial.

[2] See Lari and Young (1990) and Jelinek, Lafferty, and Mercer (1990) for introductions.

[3] CLP provides one possible approach to an operational treatment of various purely declarative grammar frameworks by an embedding of arbitrary logical languages into constraint logic programs. CLGs thus are simply understood as grammars formulated by means of a suitable logical language which can be embedded as a constraint language into a CLP scheme. Examples for an embedding of feature-based logical languages into the CLP scheme of Höhfeld and Smolka (1988) are the approaches of Dörre and Dorna (1993) and Götz (1995).

a context-sensitivity problem in that incompatible variable bindings can lead to failure derivations in dependence of the (simultaneous) history of the derivation process. Eisele (1994), Miyata (1996) and Osborne and Briscoe (1997), who attempt to adapt Baum's maximization technique to estimate the parameters of their stochastic constraint-based models, try to escape from this problem by redefining the derivation process of their respective probabilistic processing model to include only successful derivations and by renormalizing the probability distribution over derivations. Unfortunately, this move contradicts the basic independency assumption made in the parameter estimation algorithm and prohibits an application of Baum's technique as an optimization algorithm in maximum likelihood estimation of stochastic CLP.

To date to our knowledge there is no approach which solves problems 1 and 2 satisfactorily for a probabilistic model of CLP. However, an excellent starting point is the approach to "stochastic attribute-value grammars" of Abney (1996). Abney presents a probabilistic model of grammars which produce analyses in form of dags (directed acyclic graphs) by defining the probability distribution over these dags as a random field. For such probability models algorithms to estimate parameters from complete data exist (Della Pietra, Della Pietra, and Lafferty 1995) and are shown to be applicable to the stochastic grammar model. However, complete data means large corpora of costly manually analyzed, i.e., hand-parsed, data. So one open question is how to estimate parameters from incomplete, unanalyzed input, i.e., from simple corpora of natural language strings. Furthermore, if the intended application is ambiguity resolution, a second question is how to use the structure of the probabilistic model to guide the search for the most probable analysis of a string rather than simply listing all possible analyses and choosing the best one.

The aim of this paper is to present a probabilistic model of CLP and to couple this with an algorithm to induce the parameters and properties of such models from incomplete data and with an algorithm to search for best analyses. Our approach to probabilistic CLP is based on a log-linear probability model, i.e., a powerful exponential probability model well-known in probabilistic network modeling (Geman and Geman 1984; Ackley and Hinton 1985; Pearl 1988). On top of this probability model we define an algorithm to estimate the parameters and to select the properties of log-linear models from incomplete data. This induction algorithm is an extension of the improved iterative scaling algorithm of Della Pietra, Della Pietra, and Lafferty (1995) adjusted to incomplete data. The techniques developed in this context apply to log-linear models in general and are accompanied with suitable Monte Carlo approximations when applied to large data spaces. For the intended CLP application we build upon the CLP scheme of Höhfeld and Smolka (1988). In this context we present an algorithm to search for most probable analyses of the probabilistic CLP model. This algorithm is formulated as a probabilistic version of Earley deduction and can be applied to the ambiguity resolution problem in natural language processing applications[4].

The remainder of this paper is organized as follows. Section 2 introduces the basic formal concepts of CLP. Section 3 discusses in more detail the above-mentioned context-sensitivity problem in case of parameter estimation from incomplete data. Section 4 presents a general log-linear model for probabilistic CLP. Problem 1, i.e., parameter estimation of log-linear probability models from incomplete data is treated in Sect. 5. An algorithm for automatically selecting properties of log-linear models in the presence of incomplete data is also presented. Section 6 discusses the problem of estimating the terms in the formula presented in Sect. 5 in the presence of large sample spaces by Monte Carlo methods. Problem 2, i.e., methods to search for most probable analyses for probabilistic CLP, is approached in Sect. 7 in the form of a probabilistic version of Earley deduction. Section 8 gives some concluding remarks and discusses the relation of probabilistic CLP to other probabilistic processing models.

2. Constraint Logic Programming

In the following we will quickly report the basic definitions of the CLP scheme of Höhfeld and Smolka (1988). This scheme is a powerful extension of conventional logic programming (see Lloyd (1987)) and also of the CLP scheme of Jaffar and Lassez (1986) by an incorporation of arbitrary constraint languages and corresponding constraint solving methods into logic programming languages.

For example, Prolog is obtained by employing equations between first order terms as constraint language and by interpreting these equations in

[4]Even if a solution to our two problems can be seen as a necessary prerequisite for further applications such as grammar induction or language modelling, it is a necessary *and* sufficient prerequisite only for the application of ambiguity resolution. For the application of grammar induction, the question of how to impose useful constraints on the form of possible analyses in order to reduce the number of parameters to be estimated will become important. In language modelling applicatons, a shift of focus from imposing a probability distribution over a given set of analyses to imposing a probability distribution over input strings is made.

the Herbrand universe. The corresponing operational semantics of SLD-resolution can be seen to rely on a constraint solver which solves term equations in the Herbrand universe by term unification.

A constraint logic program $\mathcal{P}$ is then defined with respect to an implicit basic constraint language $\mathcal{L}$ and its relational extension $\mathcal{R}(\mathcal{L})$ as follows (see Höhfeld and Smolka (1988)).

Definition 1 (definite clause specification). *A definite clause specification $\mathcal{P}$ in $\mathcal{R}(\mathcal{L})$ is a set of definite clauses of the form*

$$A \leftarrow B_1 \,\&\, \ldots \,\&\, B_n \,\&\, \phi$$

where $A, B_1, \ldots, B_n$ are $\mathcal{R}(\mathcal{L})$-atoms, $r(\vec{x})$ is an $\mathcal{R}(\mathcal{L})$-atom iff $r \in \mathcal{R}$ is a relational symbol with arity n and $\vec{x}$ is an n-tuple of pairwise distinct variables, and ϕ is an $\mathcal{L}$-constraint ranging over the variables mentioned.

Constraint languages have to be closed under variable renaming, closed under intersection, and the satisfiability problem of such languages has to be decidable.

A **goal** is defined as a possibly empty conjunction of $\mathcal{L}$-constraints and $\mathcal{R}(\mathcal{L})$-atoms. Relying on conventional logical terminology, a **$\mathcal{P}$-answer** of a goal G for a program $\mathcal{P}$ can be defined as a satisfiable $\mathcal{L}$-constraint ϕ s.t. the implication $\phi \rightarrow G$ is a logical consequence of $\mathcal{P}$.

SLD-resolution is generalized by performing goal reduction only on the $\mathcal{R}(\mathcal{L})$-atoms and solving conjunctions of collected $\mathcal{L}$-constraints by the $\mathcal{L}$-constraint solver. **Goal reduction** is managed by a binary relation $\xrightarrow{r}$ on the set of goals as follows (V denotes the finite set of variables in the query and $\mathsf{V}(\cdot)$ is a function assigning to a constraint the finite set of variables constrained by it).

$$A \,\&\, G \xrightarrow{r} F \,\&\, G \textit{ if } A \leftarrow F \textit{ is a variant of a clause in } \mathcal{P}$$
$$\textit{s.t. } (\mathsf{V} \cup \mathsf{V}(G)) \cap \mathsf{V}(F) \subseteq \mathsf{V}(A).$$

A second rule takes care of constraint solving for the $\mathcal{L}$-constraints appearing in subsequent goals. The rule takes the conjunction of the $\mathcal{L}$-constraints from the reduced goal and the applied clause and gives, via the black box of a suitable $\mathcal{L}$-constraint solver, a satisfiable $\mathcal{L}$-constraint in solved form if the conjunction of $\mathcal{L}$-constraints is satisfiable. The **constraint solving** rule can then be defined as a total function $\xrightarrow{c}$ on the set of goals as follows ($CS(\cdot)$ denotes the $\mathcal{L}$-constraint solver as a function on the set of $\mathcal{L}$-constraints).

$$\phi \,\&\, \phi' \,\&\, G \xrightarrow{c} \phi'' \,\&\, G \textit{ if } CS(\phi \,\&\, \phi') = CS(\phi'').$$

Höhfeld and Smolka (1988) show that this generalized SLD-resolution method is a sound and complete method for inferring $\mathcal{P}$-answers. For the following discussion it will be convenient to view this operational semantics as a search of a tree. For a given query and a given program, the search space determined by the derivation rules $\xrightarrow{r}$ and $\xrightarrow{c}$ can be described as a derivation tree as follows.

Definition 2 (derivation tree). *A derivation tree determined by a query G_1 and a definite clause specification $\mathcal{P}$ has to satisfy the following conditions:*

1. *Each node is either a relation-node or a constraint-node.*
2. *The descendants of every relation-node are all constraint-nodes s.t. for every $\xrightarrow{r}$ -resolvent G' obtainable by a clause C from goal G in a relation-node, there is a descending constraint-node labeled by C and G'.*
3. *The descendants of every constraint-node are all relation-nodes s.t. for every unique $\xrightarrow{c}$ -resolvent $G\&\phi''$ obtainable from goal $G\&\phi\&\phi'$ in a constraint-node, there is a descending relation-node labeled by $G\,\&\,\phi''$.*
4. *The root node is a relation-node labeled by G_1.*
5. *A success node is a terminal relation-node labeled by a satisfiable $\mathcal{L}$ -constraint.*

Successful derivations correspond to certain subtrees of derivation trees and can be defined as proof trees as follows.

Definition 3 (proof tree). *A proof tree for a query G_1 from $\mathcal{P}$ is a subtree of a derivation tree determined by G_1 and $\mathcal{P}$ and is defined as follows:*

1. *A relation-node of the proof tree is a relation-node of the supertree and takes* one *of the descendants of the supertree relation-node as its descendant.*
2. *A constraint-node of the proof tree is a constraint-node of the supertree and takes the unique descendant of the supertree constraint-node as its descendant.*
3. *The root node of the proof tree is the root node of the supertree.*
4. *The terminal node of the proof tree is a success node of the supertree labeled by a satisfiable $\mathcal{L}$ -constraint, called answer constraint.*

3. Baum's Maximization Technique and Probabilistic CLP

One straightforward way to add statistical information to symbol processing models is to define the derivation process of such models as a a stochastic process as follows: Make a stochastic choice at each derivation step and assume the stochastic choices to be independent of each other.

Calculate the probability of a derivation as the joint probability of the independent stochastic choices made and the probability of an input as the sum of the probabilities of its derivations. This is the probabilistic model underlying, e.g., Hidden Markov Models or stochastic context-free grammars. The parameters of such models, i.e., the probabilities of the stochastic choices, can be estimated by Baum's maximization technique (Baum and Eagon 1967; Baum, Petrie, Soules, and Weiss 1970; Baum 1972).

The basic formal concepts of this technique can be described in an abstract way as follows: Let $\Pi = \{\pi_{ij}\}$ be the parameter set of an abstract probabilistic symbol processing model where $\pi_{ij} \geq 0$ and $\sum_j \pi_{ij} = 1$. The variable i ranges over the types of choices that the stochastic process makes and the variable j ranges over the alternatives to choose from when a choice of type i is made. Furthermore, let y denote an input of the probabilistic processing model, i.e., an observation sequence, and let x denote an output of the model, i.e., an analysis, and let $Y(x) = y$ be the unique observation corresponding to analysis x and $X(y) = \{x|Y(x) = y\}$ be the set of analyses of observation y. Finally, let $\nu_{ij}(x)$ be the number of selections of alternative j for a choice of type i in analysis x. Then the probability of an analysis can be calculated as the product of the probabilities of the stochastic choices made in producing it:

$$p(x; \pi) = \prod_{ij} \pi_{ij}^{\nu_{ij}(x)}$$

The probability of an observation then is the sum of the probabilities of its analyses:

$$p(y; \pi) = \sum_{x \in X(y)} p(x; \pi)$$

The purpose of Baum's maximization technique is to find maximum likelihood parameter values, i.e., $\{\pi_{ij}\}$ which maximize the likelihood function $P(\pi) = \prod_y p(y; \pi)$ for a given y-sample. To this end Baum defines a transformation τ of π into itself, which looks in its basic form as follows:

$$\tau(\pi_{ij}) = \frac{\sum_y N_{ij}}{\sum_y \big(\sum_k N_{ik}\big)} = \frac{\sum_y \big(\sum_x p(x|y)\nu_{ij}(x)\big)}{\sum_y \big(\sum_k \big(\sum_x p(x|y)\nu_{ik}(x)\big)\big)},$$

and τ yields an iterative algorithm where each step is defined by

$$\pi_{ij}^{t+1} = \tau(\pi_{ij}^t).$$

This algorithm is hill-climbing, i.e., it can be shown that $P(\tau(\pi)) > P(\pi)$ unless $\tau(\pi)$ is a critical point of P or equivalently is a fixed point of τ.

Attempts to apply this algorithm and the underlying abstract model directly to a model of probabilistic CLGs or CLP were presented, e.g., by Eisele (1994), Miyata (1996) and Osborne and Briscoe (1997). A detailed critique of such attempts with respect to the problem of parameter estimation from complete data can be found in Abney (1996). What Abney calls the Expected Rule Frequency (ERF) parameter estimation method can be seen as a special case of Baum's maximization technique. In order to show that Baum's general algorithm fails as an optimization technique for the maximum likelihood problem for probabilisitic CLP, we simply can give a counterexample using a deterministic program. In this case, parameter estimation from incomplete data using Baum's method is equivalent to using the ERF method. This point shall be made explicit in the following.

Let us apply the above-defined abstract model to a simple deterministic constraint logic program. The stochastic choices of the abstract model correspond to application probabilities of definite clauses in the generalized SLD-resolution procedure; the alternatives to choose from when an atom is selected in goal reduction are the different clauses defining the selected atom. In the following example (see Fig. 1), each clause will be annotated by a choice-alternative pair indicating a probabilisitic parameter π_{ij}.

$$
\begin{array}{l}
\texttt{11}\ s(Z) \leftarrow p(Z)\ \&\ q(Z). \\
\texttt{21}\ p(Z) \leftarrow Z = a. \\
\texttt{22}\ p(Z) \leftarrow Z = b. \\
\texttt{31}\ q(Z) \leftarrow Z = a. \\
\texttt{32}\ q(Z) \leftarrow Z = b.
\end{array}
$$

FIGURE 1. A sample program

The relational atom $s(Z)$ is defined uniquely in clause `11`. The atoms $p(Z)$ and $q(Z)$ each are defined in two different ways, which for the sake of the example are considered to be incompatible. For a selection of atom $p(Z)$ one can choose between clauses `21` and `22` in a goal reduction step, whereas for a choice of atom $q(Z)$ the alternatives to choose from are clauses `31` and `32`. This program is deterministic for the queries $s(Z)\&Z = a$ and $s(Z)\ \&\ Z = b$. This means, there is only one proof tree from the above program for each query (see Fig. 2). The proof tree x_1 for the query $s(Z)\ \&\ Z = a$ uses clauses `11, 21` and `31` and yields answer constraint $Z = a$; the proof tree x_2 for the second query uses clauses `11, 22` and `32` and gives answer constraint $Z = b$.

Let us now consider the application of Baum's maximization technique to estimate the parameters of such a probabilistic CLP model (see Fig. 3). An input corpus consisting of the three queries $y_1 : s(Z)\ \&\ Z = a$, $y_2 : s(Z)\ \&\ Z = a$ and $y_3 : s(Z)\ \&\ Z = b$ will yield the corresponding unique proof trees $x_1 \in X(y_1)$, $x_1 \in X(y_2)$ and $x_2 \in X(y_3)$. The

$$x_1: \quad \begin{array}{c} s(Z)\ \&\ Z=a \\ \big|\ r,c \\ \mathtt{11},\ p(Z)\ \&\ q(Z)\ \&\ Z=a \\ \big|\ r,c \\ \mathtt{21},\ q(Z)\ \&\ Z=a \\ \big|\ r,c \\ \mathtt{31},\ Z=a \end{array}$$

$$x_2: \quad \begin{array}{c} s(Z)\ \&\ Z=b \\ \big|\ r,c \\ \mathtt{11},\ p(Z)\ \&\ q(Z)\ \&\ Z=b \\ \big|\ r,c \\ \mathtt{22},\ q(Z)\ \&\ Z=b \\ \big|\ r,c \\ \mathtt{32},\ Z=b \end{array}$$

FIGURE 2. Proof trees from sample program

y	$x \in X(y)$	$p(x\|y)$	N_{11}	N_{21}	N_{22}	N_{31}	N_{32}
y_1	x_1	1	$1 \cdot 1$	$1 \cdot 1$	$1 \cdot 0$	$1 \cdot 1$	$1 \cdot 0$
y_2	x_1	1	$1 \cdot 1$	$1 \cdot 1$	$1 \cdot 0$	$1 \cdot 1$	$1 \cdot 0$
y_3	x_2	1	$1 \cdot 1$	$1 \cdot 0$	$1 \cdot 1$	$1 \cdot 0$	$1 \cdot 1$
		$\sum_y N_{ij} =$	3	2	1	2	1
		$\sum_y \sum_k N_{ik} =$	3	3	3	3	3
		$\hat{\pi}_{ij} =$	1	2/3	1/3	2/3	1/3

FIGURE 3. A sample estimation

conditional probabilities $p(x|y)$ for $x \in X(y)$ will be 1 in each case since there is a unique proof tree for each query. Thus for the calculation of $N_{ij} = \sum_x p(x|y)\nu_{ij}(x)$, the expected number of occurences of clauses in proof trees, we simply have to count and can ignore the respective probabilities of the proof trees. The algorithm then will give unique estimated parameter values $\hat{\pi}_{ij} = \frac{\sum_y N_{ij}}{\sum_y \left(\sum_k N_{ik}\right)}$ immediately.

If we now consider the calculation of the probability distribution over the proof trees of such a probabilistic CLP model, we see that in contrast to the above-defined abstract model we cannot simply calculate a product for each proof tree. Instead, in order to get a proper probability distribution over proof trees, we have to do an additional normalization. For example, if the sum of the unnormalized probabilities of the proof trees under the estimated model, $p(x_1;\hat{\pi}) + p(x_2;\hat{\pi}) = 4/9 + 1/9 = 5/9$, is used as a normalization constant, then we will get a normalized probability distribution over proof trees, $p'(x_1;\pi') = 4/5$, $p'(x_2;\pi') = 1/5$, yielding a normalized likelihood of our training corpus $P'(\pi') = (4/5)^2 \cdot 1/5 = .128$. Note that the normalized probability distribution no longer refers to specific parameter values. In fact, there is no analytical solution to the problem of finding parameter values π' for the program of Fig. 1 which yield probability distribution p' over the proof trees of Fig. 2. However, given the same preconditions, we can find a probability distribution $p''(x_1;\pi'') = 2/3$, $p''(x_2;\pi'') = 1/3$ which yields a higher likelihood $P''(\pi'') = (2/3)^2 \cdot 1/3 = .148$. This contradicts the assumption that the parameter values estimated by Baum's technique are the requested maximum likelihood estimates for a probabilistic CLP model as defined above.

4. A Log-linear Model for Probabilistic CLP

The above-discussed approach based a probability distribution over proof trees on a definition of the derivation process of CLP as a (context-free) stochastic process. An alternative, presented by Abney (1996), for his model of stochastic attribute-value grammars is to define a probability distribution over dags as a random field. This probability model does not build on any underlying stochastic process but rather on the underlying graphical structure of the analyses produced by the model. Random fields can be seen as special instances of general log-linear probability models. Such a model can be defined as follows.

Definition 4 (log-linear distribution)**.** *A log-linear probability distribution $p_{\lambda\cdot\nu}$ on a set Ω is defined s.t. for all $\omega \in \Omega$:*

$$p_{\lambda\cdot\nu}(\omega) = Z_{\lambda\cdot\nu}{}^{-1} e^{\lambda\cdot\nu(\omega)} p_0(\omega)$$

where $Z_{\lambda\cdot\nu} = \sum_{\omega\in\Omega} e^{\lambda\cdot\nu(\omega)} p_0(\omega)$ *is a normalizing constant,*
$\lambda = (\lambda_1, \dots, \lambda_n)$ *is a vector of log-parameters s.t.* $\lambda \in \mathbb{R}^n$,
$\chi = (\chi_1, \dots, \chi_n)$ *is a vector of properties,*
$\nu = (\nu_1, \dots, \nu_n)$ *is a vector of property-functions s.t. for each*
$\nu_i : \Omega \to \mathbb{N}$, $\nu_i(\omega)$ *is the number of occurences of property* χ_i *in* ω,
$\lambda \cdot \nu(\omega)$ *is a weighted property-function s.t.* $\lambda \cdot \nu(\omega) = \sum_{i=1}^{n} \lambda_i \nu_i(\omega)$,
p_0 *is a fixed initial distribution.*

In analogy to stochastic attribute-values grammars, we can define a probability distribution over proof trees as a special log-linear model. The special instance of interest is simply a log-linear distribution on the countably infinite set of proof-trees for a set of queries to a program. Such a distribution is determined by a vector of properties and a corresponding vector of log-parameters. Properties could be defined, e.g., as subtrees of proof trees. For the moment, we can leave an exact definition of properties aside and refer to an assumed vector of property-functions.

The form of log-linear models can be rationalized as an example of an **exponential family** of probability functions. From this viewpoint this model can be seen as just a very flexible probability model defining the probability of a configuration to be proportional to the product of weights assigned to arbitrary properties of the configuration.

$$p(\omega) \propto \prod_{i=1}^{n} \pi_i^{\nu_i(\omega)}.$$

This can be put in the form of Definition 4 by replacing proportionality by a constant and parameters π_i by log-parameters $\lambda_i = log\ \pi_i$.

$$\begin{aligned} p(\omega) &= C \prod_{i=1}^{n} \pi_i^{\nu_i(\omega)} \\ &= Z^{-1} \prod_{i=1}^{n} \pi_i^{\nu_i(\omega)} \\ &= Z^{-1} \prod_{i=1}^{n} e^{\lambda_i \nu_i(\omega)} \\ &= Z^{-1} e^{\sum_{i=1}^{n} \lambda_i \nu_i(\omega)}. \end{aligned}$$

Another way to rationalize the form of the log-linear model is as a **maximum entropy** probability distribution. From this viewpoint we do statistical inference and, believing that entropy is the unique consistent measure of the amount of uncertainty represented by a probability distribution, we obey the following principle:

> In making inferences on the basis of partial information we must use that probability distribution which has maximum entropy subject to whatever is known. This is the only unbiased assignment we can make; to use any other would amount to arbitrary assumption of information which by hypothesis we do not have. (Jaynes 1957)

More formally, suppose a random variable X can take on values $x_i, i = 1, \ldots, n$ and we want to estimate the corresponding probabilities $p_i, i = 1, \ldots, n$. All we have are expectations of functions $f_k(X), k = 1, \ldots, m$. The maximum entropy principle can then be stated as follows.

> *Maximize* $H(p_1, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log p_i$ *subject to the constraints* $\sum_{i=1}^{n} p_i f_k(x_i) = F_k, k = 1, \ldots, m$ *and* $\sum_{i=1}^{n} p_i = 1$.

The solution we get for all $p_i, i = 1, \ldots, n$ is:

$$p_i = \frac{e^{\sum_{k=1}^{m} \lambda_k f_k(x_i)}}{\sum_{i=1}^{n} e^{\sum_{k=1}^{m} \lambda_k f_k(x_i)}}.$$

This result follows directly from a constrained optimization argument where the parameters are viewed as Lagrange multipliers:

Let Λ denote the Lagrangian defined by $\Lambda(p_1, \ldots, p_n, \lambda_0, \lambda_1, \ldots, \lambda_m) = -\sum_{i=1}^{n}(p_i \, log \, p_i) + (\lambda_0 + 1)\sum_{i=1}^{n}(p_i - 1) + \lambda_1 \sum_{i=1}^{n}(p_i f_1(x_i) - F_1) + \cdots + \lambda_m \sum_{i=1}^{n}(p_i f_m(x_i) - F_m)$.
Then $\frac{\partial}{\partial p_i}\Lambda = -(log \; p_i + 1) + (\lambda_0 + 1) + \lambda_1 f_1(x_i) + \cdots + \lambda_m f_m(x_i)$.
Set $\frac{\partial}{\partial p_i}\Lambda = 0$, then $p_i = e^{\lambda_0 + \sum_{k=1}^{m} \lambda_k f_k(x_i)}$.
Since $\sum_{i=1}^{n} p_i = 1$, we have $e^{\lambda_0} \sum_{i=1}^{n} e^{\sum_{k=1}^{m} \lambda_k f_k(x_i)} = 1$.
Define $Z = \sum_{i=1}^{n} e^{\sum_{k=1}^{m} \lambda_k f_k(x_i)}$, then $\lambda_0 = log \; Z^{-1}$
and $p_i = Z^{-1} e^{\sum_{k=1}^{m} \lambda_k f_k(x_i)} = \frac{e^{\sum_{k=1}^{m} \lambda_k f_k(x_i)}}{\sum_{i=1}^{n} e^{\sum_{k=1}^{m} \lambda_k f_k(x_i)}}$.

Log-linear models originated in statistical physics as flexible probabilistic models of equilibrium states of physical systems. Jaynes interpreted such Gibbs- or Boltzmann-distributions in a more abstract maximum-entropy framework (see Jaynes (1983)). Besides numerous applications in the area of natural language processing[5], log-linear models are also applied successfully in image processing (see the work on random fields initiated by Geman and Geman (1984)) and are closely related to other probabilistic models such as Boltzmann machines (see Ackley and Hinton (1985)) or graphical models (see Pearl (1988)).

The work presented in the following sections applies for the most part to log-linear models in general. We will refer for this discussion to Definition 4. In case the property vector is fixed and clear from the context, the model will be written p_λ to indicate the depencence on the parameter vector. Furthermore, it will be convenient to have a recursive definition of models based on property-functions which are extended by additional properties and corresponding parameters or by new parameters.

Proposition 1. *For each weighted property-funtion* $\phi(\omega) = \lambda \cdot \nu(\omega)$, $\psi(\omega) = \gamma \cdot \mu(\omega)$ *(with possibly* $\nu = \mu$*), let* $(\psi + \phi)(\omega) = \psi(\omega) + \phi(\omega)$ *be an extended property-function (reducing to* $\lambda + \gamma$ *in case* $\nu = \mu$*). Then* $p_{\psi+\phi}(\omega) = {Z_{\psi+\phi}}^{-1} e^{\psi(\omega)} p_\phi(\omega)$ *where* $Z_{\psi+\phi} = \sum_{\omega \in \Omega} e^{\psi(\omega)} p_\phi(\omega)$.

[5]The applications include, beside others, probabilistic grammar models (Mark, Miller, Grenander, and Abney 1992; Abney 1996), word morphology (Della Pietra, Della Pietra, and Lafferty 1995), machine translation (Berger, Della Pietra, and Della Pietra 1996), language modelling (Rosenfeld 1996), part-of-speech tagging (Ratnaparkhi 1996), word correlations (Beeferman, Berger, and Lafferty 1997a) and text segmentation (Beeferman, Berger, and Lafferty 1997b).

Proof.

$$\begin{aligned}
p_{\psi+\phi}(\omega) &= Z_{\psi+\phi}{}^{-1} e^{\psi+\phi(\omega)} p_0(\omega) \\
&= (\sum_{\omega \in \Omega} e^{\psi(\omega)+\phi(\omega)} p_0(\omega))^{-1} e^{\psi(\omega)+\phi(\omega)} p_0(\omega) \\
&= (\sum_{\omega \in \Omega} e^{\psi(\omega)} e^{\phi(\omega)} p_0(\omega) Z_\phi Z_\phi{}^{-1})^{-1} e^{\psi(\omega)} e^{\phi(\omega)} p_0(\omega) \\
&= Z_\phi{}^{-1} (\sum_{\omega \in \Omega} e^{\psi(\omega)} p_\phi(\omega))^{-1} e^{\psi(\omega)} e^{\phi(\omega)} p_0(\omega) \\
&= (\sum_{\omega \in \Omega} e^{\psi(\omega)} p_\phi(\omega))^{-1} e^{\psi(\omega)} p_\phi(\omega). \quad \square
\end{aligned}$$

5. Inducing Log-linear Models from Incomplete Data

Induction of log-linear models involves two problems: parameter estimation and property selection. In the following we will give a detailed presentation of solutions to these problems for the case of incomplete data.

5.1. Parameter Estimation from Incomplete Data.

An algorithm to estimate the parameters of general log-linear models from complete data has been presented by Della Pietra, Della Pietra, and Lafferty (1995). Their "Improved Iterative Scaling" algorithm is an extension of the "Generalized Iterative Scaling" algorithm of Darroch and Ratcliff (1972) especially tailored to estimating models with large parameter spaces. The algorithm is a technique for maximum likelihood estimation for log-linear models from complete data, i.e., it addresses the problem of maximizing the complete-data log-likelihood function $log \prod_x p(x)^{\tilde{p}(x)}$ for a given empirical distribution $\tilde{p}(x)$ over complete data x. The solution to this problem is equivalent to the solution to the maximum entropy problem subject to linear constraints, i.e., the problem of maximizing the entropy $H(p)$ subject to the constraints $\sum_x p(x) f_k(x) = \sum_x \tilde{p}(x) f_k(x),\ k = 1, \dots, m$ with respect to the complete data empirical expectation (see Della Pietra, Della Pietra, and Lafferty (1995)). In the language of constrained optimization, the maximum likelihood problem for log-linear models with respect to complete data is the dual to the maximum entropy problem for linear constraints with respect to complete data (see Berger, Della Pietra, and Della Pietra (1996)).

However, the need to rely on large training samples of complete data may be inconvenient if complete data are complex and difficult to gather.

This is the case for applications of CLP to natural language processing. Here complete data means several person-years of hand-annotating large corpora with detailed analyses of specialized grammar frameworks. Clearly, for such applications parameter estimation from incomplete data, i.e., unanalyzed input of natural language strings, is desirable.

Unfortunately, Iterative Scaling will no longer work if the training data are incomplete. The incomplete-data log-likelihood takes the form $log \prod_y \sum_{x \in X(y)} p(x)$, i.e., the probability the model assigns to the data strings is the product of the probabilities of the strings and the probability of a string is calculated as the sum of the probabilities of its analyses. In contrast to the complete-data log-likelihood this function is non-concave (it involves a sum inside the logarithm) and cannot be maximized directly or uniquely.

In the following we will show how the numerical algorithm of Della Pietra, Della Pietra, and Lafferty (1995) can be redefined in order to fit incomplete data. The new algorithm can be defined in the EM-framework of maximum likelihood estimation from incomplete data of (Dempster, Laird, and Rubin 1977). Applying this framework to the problem of probabilistic CLP, we can assume the following to be given:

- *Observed, incomplete data* $y \in \mathcal{Y}$ *corresponding to a given, finite set of queries for a constraint logic program* $\mathcal{P}$ *,*
- *Unobserved, complete data* $x \in \mathcal{X}$ *corresponding to the countably infinite set of proof trees for queries* $\mathcal{Y}$ *from a constraint logic program* $\mathcal{P}$ *,*
- *Functions* $Y : \mathcal{X} \to \mathcal{Y}$ *s.t.* $Y(x) = y$ *corresponds to the unique query labeling proof tree* x*, and* $X : \mathcal{Y} \to \mathcal{X}$ *s.t.* $X(y) = \{x | \, Y(x) = y\}$ *is the countably infinite set of proof trees for query* y *from a constraint logic program* $\mathcal{P}$ *,*
- *Complete data specifications* p_λ *s.t.* $p_\lambda(x)$ *is a log-linear distribution on* $\mathcal{X}$ *with given initial distribution* p_0*, fixed properties* χ *and property-functions vector* ν *and depending on parameter vector* λ*,*
- *Incomplete data specifications* L *s.t.* $L(\lambda) = log \prod_{y \in \mathcal{Y}} \sum_{x \in X(y)} p_\lambda(x)$ $= \sum_{y \in \mathcal{Y}} log \sum_{x \in X(y)} p_\lambda(x) = \sum_{y \in \mathcal{Y}} log \; p_\lambda(y)$ *is the log-likelihood of a fixed* $\mathcal{Y}$*-sample depending on parameter vector* λ*.*

For the discussion of parameter estimation we will refer to a given vector of property functions. This is assumed to result from the property selection procedure defined in Sect. 5.2, whereby for each property function ν_i some proof tree $x \in \mathcal{X}$ s.t. $\nu_i(x) > 0$ is assumed to exist. Furthermore, we require p_λ to be strictly positive on $\mathcal{X}$, i.e., $p_\lambda(x) > 0$ for all $x \in \mathcal{X}$.

The problem of maximum likelihood estimation of log-linear models from incomplete data can then be stated formally as follows.

> *Given a fixed $\mathcal{Y}$-sample and a set $\Lambda = \{\lambda|\ p_\lambda(x)$ is a log-linear distribution on $\mathcal{X}$ with fixed p_0, fixed ν and $\lambda \in \mathrm{IR}^n\}$, we want to find the maximum likelihood estimate $\lambda^* \in \Lambda$ s.t. $\lambda^* = argmax_{\lambda \in \Lambda} L(\lambda)$.*

The key idea of the following approach is to iteratively maximize a strictly concave auxiliary function when the log-likelihood objective function cannot be maximized analytically. An auxiliary function convenient for our problem can be defined as a two-place function A giving an estimate of the improvement in the incomplete-data log-likelihood L when going from a model p_λ to a model $p_{\gamma+\lambda}$.

In the following $p[f] = \sum_{\omega \in \Omega} p(\omega) f(\omega)$ will denote the expectation of a function $f : \Omega \to \mathrm{IR}$ with respect to a probability distribution p on a set Ω.

Definition 5. *Let $\lambda \in \Lambda$, $\gamma \in \mathrm{IR}^n$. Then*

$$A(\gamma + \lambda) = \sum_{y \in \mathcal{Y}} (1 + k_\lambda [\gamma \cdot \nu] - p_\lambda [\sum_{i=1}^n \bar{\nu}_i e^{\gamma_i \nu_\#}])$$

where $\bar{\nu}_i(x) = \frac{\nu_i(x)}{\nu_\#(x)}$, $\nu_\#(x) = \sum_{i=1}^n \nu_i(x)$, $k_\lambda(x) = \frac{p_\lambda(x)}{\sum_{x \in X(y)} p_\lambda(x)}$.

By considering the first and second derivative of A, we see that A is strictly concave in the parameters. Strict concavity together with continuity of the function and closedness of the parameter space directly gives us a unique maximum of A.

Proposition 2. *For each $\lambda \in \Lambda$, $\gamma \in \mathrm{IR}^n$: $A(\gamma+\lambda)$ takes its maximum as a function of γ at the unique point $\hat{\gamma}$ satisfying for each $\hat{\gamma}_i, i = 1, \ldots, n$:*

$$\sum_{y \in \mathcal{Y}} k_\lambda [\nu_i] = \sum_{y \in \mathcal{Y}} p_\lambda \left[\nu_i e^{\hat{\gamma}_i \nu_\#}\right].$$

Proof.

$$\begin{aligned}
\frac{\partial}{\partial \gamma_i} A(\gamma + \lambda) &= \frac{\partial}{\partial \gamma_i} \sum_{y \in \mathcal{Y}} (1 + k_\lambda[\gamma \cdot \nu] - p_\lambda[\sum_{j=1}^n \bar{\nu}_j e^{\gamma_j \nu_\#}]) \\
&= \sum_{y \in \mathcal{Y}} (\frac{\partial}{\partial \gamma_i} \sum_{j=1}^n (\frac{1}{n} + k_\lambda[\gamma_j \cdot \nu_j] - p_\lambda[\bar{\nu}_j e^{\gamma_j \nu_\#}])) \\
&= \sum_{y \in \mathcal{Y}} (\sum_{j \neq i} (\frac{\partial}{\partial \gamma_i} (\frac{1}{n} + k_\lambda[\gamma_j \cdot \nu_j] - p_\lambda[\bar{\nu}_j e^{\gamma_j \nu_\#}])) \\
&\quad + \frac{\partial}{\partial \gamma_i} (\frac{1}{n} + k_\lambda[\gamma_i \cdot \nu_i] - p_\lambda[\bar{\nu}_i e^{\gamma_i \nu_\#}]))
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{y \in \mathcal{Y}} (k_\lambda[\nu_i] - \sum_{x \in \mathcal{X}} (p_\lambda(x) \bar{\nu}_i(x) e^{\gamma_i \nu_\#(x)} \nu_\#(x))) \\
&= \sum_{y \in \mathcal{Y}} (k_\lambda[\nu_i] - \sum_{x \in \mathcal{X}} (p_\lambda(x) \nu_i(x) e^{\gamma_i \nu_\#(x)})) \\
&= \sum_{y \in \mathcal{Y}} (k_\lambda[\nu_i] - p_\lambda[\nu_i e^{\gamma_i \nu_\#}]).
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial^2}{\partial {\gamma_i}^2} A(\gamma + \lambda) &= \frac{\partial}{\partial \gamma_i} \sum_{y \in \mathcal{Y}} (k_\lambda[\nu_i] - p_\lambda[\nu_i e^{\gamma_i \nu_\#}]) \\
&= - \sum_{y \in \mathcal{Y}} (\frac{\partial}{\partial \gamma_i} p_\lambda[\nu_i e^{\gamma_i \nu_\#}]) \\
&= - \sum_{y \in \mathcal{Y}} (\sum_{x \in \mathcal{X}} (p_\lambda(x) \nu_i(x) e^{\gamma_i \nu_\#(x)} \nu_\#(x))) \\
&= - \sum_{y \in \mathcal{Y}} p_\lambda[\nu_i \nu_\# e^{\gamma_i \nu_\#}] \\
&< 0. \quad \square
\end{aligned}
$$

At the core of the proposed method lies the definition of an iterative algorithm for maximizing L which is constructed from the auxiliary function A. At each step of this "Iterative Maximization (IM)" algorithm a model based on parameters λ is extended by a parameter vector $\hat{\gamma}$ which gives the maximum estimated improvement in log-likelihood L, i.e., which is obtained by maximizing the auxiliary function $A(\gamma + \lambda)$ as a function of γ.

Definition 6 (iterative maximization)**.** *Let $\mathcal{M} : \Lambda \to \Lambda$ be a mapping defined by*

$$\mathcal{M}(\lambda) = \hat{\lambda} \in \Lambda \ s.t. \ \hat{\lambda} = \hat{\gamma} + \lambda \ with \ \hat{\gamma} = argmax_{\gamma \in \mathbb{R}^n} A(\gamma + \lambda).$$

Then each step of the Iterative Maximization Algorithm is defined by

$$\lambda^{(k+1)} = \mathcal{M}(\lambda^{(k)}).$$

To show the central convergence properties of the IM algorithm, we first have to show some provisional results. Lemma 3 shows that the auxiliary function $A(\gamma + \lambda)$ is a lower bound on $L(\gamma + \lambda) - L(\lambda)$, the difference in log-likelihood between the basic and the extended model, i.e., it is a conservative estimate of the improvement in log-likelihood.

Lemma 3. *$A(\gamma + \lambda) \leq L(\gamma + \lambda) - L(\lambda)$.*

Proof.

$$\begin{aligned}
L(\gamma+\lambda)-L(\lambda) &= \sum_{y\in\mathcal{Y}}(log\frac{p_{\gamma+\lambda}(y)}{p_\lambda(y)}) \\
&= \sum_{y\in\mathcal{Y}}(log\frac{1}{p_\lambda(y)}\sum_{x\in X(y)}(p_{\gamma+\lambda}(x)\frac{p_\lambda(x)}{p_\lambda(x)})) \\
&= \sum_{y\in\mathcal{Y}}(log\sum_{x\in X(y)}(\frac{p_\lambda(x)}{p_\lambda(y)}\frac{p_{\gamma+\lambda}(x)}{p_\lambda(x)})) \\
&\geq \sum_{y\in\mathcal{Y}}(\sum_{x\in X(y)}(\frac{p_\lambda(x)}{p_\lambda(y)}log\frac{p_{\gamma+\lambda}(x)}{p_\lambda(x)})) \\
&\quad \text{by Jensen's inequality} \\
&= \sum_{y\in\mathcal{Y}}(\sum_{x\in X(y)}(\frac{p_\lambda(x)}{p_\lambda(y)}(log\ p_{\gamma+\lambda}(x)-log\ p_\lambda(x)))) \\
&= \sum_{y\in\mathcal{Y}}(\sum_{x\in X(y)}(\frac{p_\lambda(x)}{p_\lambda(y)}(log\ Z^{-1}_{\gamma+\lambda}+log\ e^{\gamma\cdot\nu(x)} \\
&\quad +log\ p_\lambda(x)-log\ p_\lambda(x)))) \\
&= \sum_{y\in\mathcal{Y}}(k_\lambda[\gamma\cdot\nu]-log\ p_\lambda[e^{\gamma\cdot\nu}]) \\
&\geq \sum_{y\in\mathcal{Y}}(k_\lambda[\gamma\cdot\nu]+1-p_\lambda[e^{\gamma\cdot\nu}]) \quad \text{since } log\ x\leq x-1 \\
&= \sum_{y\in\mathcal{Y}}(k_\lambda[\gamma\cdot\nu]+1-\sum_{x\in\mathcal{X}}(p_\lambda(x)e^{\sum_{i=1}^n\gamma_i\nu_i(x)\frac{\nu_\#(x)}{\nu_\#(x)}})) \\
&= \sum_{y\in\mathcal{Y}}(k_\lambda[\gamma\cdot\nu]+1-\sum_{x\in\mathcal{X}}(p_\lambda(x)e^{\sum_{i=1}^n\gamma_i\bar{\nu}_i(x)\nu_\#(x)})) \\
&\geq \sum_{y\in\mathcal{Y}}(k_\lambda[\gamma\cdot\nu]+1-\sum_{x\in\mathcal{X}}(p_\lambda(x)\sum_{i=1}^n\bar{\nu}_i(x)e^{\gamma_i\nu_\#(x)})) \\
&\quad \text{by Jensen's inequality} \\
&= \sum_{y\in\mathcal{Y}}(k_\lambda[\gamma\cdot\nu]+1-p_\lambda[\sum_{i=1}^n\bar{\nu}_i e^{\gamma_i\nu_\#}]) \\
&= A(\gamma+\lambda). \quad \square
\end{aligned}$$

Lemma 4 shows that there is no estimated improvement in log-likelihood in the origin.

Lemma 4. *$A(0+\lambda) = 0$.*

Proof.

$$A(0+\lambda) = \sum_{y \in \mathcal{Y}} (k_\lambda[0 \cdot \nu] + 1 - \sum_{x \in \mathcal{X}} p_\lambda(x) \sum_{i=1}^{n} \bar{\nu}_i(x) e^0) = 0. \quad \square$$

Lemma 5 shows that the critical points of interest are the same for A and L.

Lemma 5. $\frac{d}{dt}\big|_{t=0} A(t\gamma+\lambda) = \frac{d}{dt}\big|_{t=0} L(t\gamma+\lambda)$.

Proof.

$$\begin{aligned}
\frac{d}{dt} A(t\gamma+\lambda) &= \frac{d}{dt} \sum_{y \in \mathcal{Y}} (k_\lambda[t\gamma \cdot \nu] + 1 - \sum_{x \in \mathcal{X}} (p_\lambda(x) \sum_{i=1}^{n} \bar{\nu}_i(x) e^{t\gamma_i \nu_\#(x)})) \\
&= \sum_{y \in \mathcal{Y}} (k_\lambda[\gamma \cdot \nu] - \sum_{x \in \mathcal{X}} (p_\lambda(x) \sum_{i=1}^{n} \frac{\nu_i(x)}{\nu_\#(x)} e^{t\gamma_i \nu_\#(x)} \gamma_i \nu_\#(x))) \\
&= \sum_{y \in \mathcal{Y}} (k_\lambda[\gamma \cdot \nu] - \sum_{x \in \mathcal{X}} (p_\lambda(x) \sum_{i=1}^{n} \nu_i(x) \gamma_i e^{t\gamma_i \nu_\#(x)})). \\
\frac{d}{dt}\Big|_{t=0} A(t\gamma+\lambda) &= \sum_{y \in \mathcal{Y}} (k_\lambda[\gamma \cdot \nu] - \sum_{x \in \mathcal{X}} (p_\lambda(x) \sum_{i=1}^{n} \nu_i(x) \gamma_i e^0)) \\
&= \sum_{y \in \mathcal{Y}} (k_\lambda[\gamma \cdot \nu] - p_\lambda[\gamma \cdot \nu]). \\
\frac{d}{dt} L(t\gamma+\lambda) &= \sum_{y \in \mathcal{Y}} (\frac{d}{dt} log \sum_{x \in X(y)} p_{t\gamma+\lambda}(x)) \\
&= \sum_{y \in \mathcal{Y}} ((\sum_{x \in X(y)} p_{t\gamma+\lambda}(x))^{-1} \frac{d}{dt} \sum_{x \in X(y)} e^{t\gamma \cdot \nu(x)} p_\lambda(x) Z^{-1}_{t\gamma+\lambda}) \\
&= \sum_{y \in \mathcal{Y}} ((\sum_{x \in X(y)} p_{t\gamma+\lambda}(x))^{-1} \sum_{x \in X(y)} p_\lambda(x) \\
&\quad (-e^{t\gamma \cdot \nu(x)} Z^{-2}_{t\gamma+\lambda} \sum_{x \in \mathcal{X}} e^{t\gamma \cdot \nu(x)} \gamma \cdot \nu(x) p_\lambda(x) \\
&\quad + Z^{-1}_{t\gamma+\lambda} e^{t\gamma \cdot \nu(x)} \gamma \cdot \nu(x))) \\
&= \sum_{y \in \mathcal{Y}} (- \sum_{x \in X(y)} p_{t\gamma+\lambda}(x) p_{t\gamma+\lambda}[\gamma \cdot \nu] (\sum_{x \in X(y)} p_{t\gamma+\lambda}(x))^{-1}
\end{aligned}$$

$$
\begin{aligned}
& + \sum_{x \in X(y)} p_{t\gamma+\lambda}[\gamma \cdot \nu] (\sum_{x \in X(y)} p_{t\gamma+\lambda}(x))^{-1}) \\
= & \sum_{y \in \mathcal{Y}} (-p_{t\gamma+\lambda}[\gamma \cdot \nu] + k_{t\gamma+\lambda}[\gamma \cdot \nu]).
\end{aligned}
$$

$$
\left.\frac{d}{dt}\right|_{t=0} L(t\gamma + \lambda) = \sum_{y \in \mathcal{Y}} (k_\lambda[\gamma \cdot \nu] - p_\lambda[\gamma \cdot \nu]). \quad \square
$$

One central result of this section is stated in Theorem 6. It shows the hill-climbing nature of the IM algorithm, i.e., the log-likelihood L is increasing on each iteration of the IM algorithm except at fixed points of $\mathcal{M}$ or equivalently at critical points of L.

Theorem 6. *For all $\lambda \in \Lambda$: $L(\mathcal{M}(\lambda)) \geq L(\lambda)$ with equality iff λ is a fixed point of $\mathcal{M}$ or equivalently is a critical point of L.*

Proof.

$$
\begin{aligned}
L(\mathcal{M}(\lambda)) - L(\lambda) & \geq A(\mathcal{M}(\lambda)) \quad \text{by Lemma 3} \\
& \geq 0 \quad \text{by Lemma 4 and definition of } \mathcal{M}.
\end{aligned}
$$

The equality $L(\mathcal{M}(\lambda)) = L(\lambda)$ holds iff λ is a fixed point of $\mathcal{M}$, i.e., $\mathcal{M}(\lambda) = \hat{\gamma} + \lambda$ with $\hat{\gamma} = 0$. Furthermore, λ is a fixed point of $\mathcal{M}$ iff $\hat{\gamma} = argmax_{\gamma \in \mathbb{R}^n} A(\gamma + \lambda) = 0$,
$\iff$ for all $\gamma \in \mathbb{R}^n$: $\hat{t} = argmax_{t \in \mathbb{R}} A(t\gamma + \lambda) = 0$,
$\iff$ for all $\gamma \in \mathbb{R}^n$: $\left.\frac{d}{dt}\right|_{t=0} A(t\gamma + \lambda) = 0$,
$\iff$ for all $\gamma \in \mathbb{R}^n$: $\left.\frac{d}{dt}\right|_{t=0} L(t\gamma + \lambda) = 0$, by Lemma 5
$\iff$ λ is a critical point of L. $\square$

Corollary 7 implies that a maximum likelihood estimate is a fixed point of the mapping $\mathcal{M}$.

Corollary 7. *Let $\lambda^* = argmax_{\lambda \in \Lambda} L(\lambda)$. Then λ^* is a fixed point of $\mathcal{M}$.*

Proposition 8 discusses the convergence properties of the algorithm. As with each application of the EM algorithm, we can show convergence of the IM algorithm to critical points of the incomplete-data log-likelihood function L. This means that the limiting parameter value can occur at a local, not only at a global maximum of L. This chaotic behaviour of the algorithm, i.e., the dependence of convergence on initial parameter values, must be treated as an empirical matter.

Proposition 8. *Let $\{\lambda^{(k)}\}$ be a sequence in Λ determined by the IM Algorithm. Then all limit points of $\{\lambda^{(k)}\}$ are fixed points of $\mathcal{M}$ or equivalently are critical points of L.*

Proof. Let $\{\lambda^{(k_n)}\}$ be a subsequence of $\{\lambda^{(k)}\}$ converging to $\bar{\lambda}$. Then for all $\gamma \in \mathbb{R}^n$:

$$\begin{aligned}
A(\gamma + \lambda^{(k_n)}) &\leq A(\hat{\gamma}^{(k_n)} + \lambda^{(k_n)}) \quad \text{by definition of } \mathcal{M} \\
&\leq L(\hat{\gamma}^{(k_n)} + \lambda^{(k_n)}) - L(\lambda^{(k_n)}) \quad \text{by Lemma 3} \\
&= L(\lambda^{(k_n+1)}) - L(\lambda^{(k_n)}) \quad \text{by definition of IM} \\
&\leq L(\lambda^{(k_{n+1})}) - L(\lambda^{(k_n)})
\end{aligned}$$

and in limit as $n \to \infty$ for continuous A and L: $A(\gamma+\bar{\lambda}) \leq L(\bar{\lambda})-L(\bar{\lambda}) = 0$. Thus $\gamma = 0$ is a maximum of $A(\gamma + \bar{\lambda})$, using Lemma 4, and $\bar{\lambda}$ is a fixed point of $\mathcal{M}$. Furthermore, $\frac{d}{dt}\big|_{t=0} A(t\gamma + \bar{\lambda}) = \frac{d}{dt}\big|_{t=0} L(t\gamma + \bar{\lambda}) = 0$, using Lemma 5, and $\bar{\lambda}$ is a critical point of L. □

5.2. **Property Selection from Incomplete Data.** For the preceding task of parameter estimation we assumed a vector of properties to be given. However, exhaustive sets of properties can get unmanageably large for most applications. Let us consider the application of probabilistic CLP: One possible definition of properties of proof trees is as subtrees of proof trees. If we want to be as flexible as possible in the definition of subtree-properties and define a subtree of a proof tree to be an arbitrary subgraph of a proof tree, then the number of subtrees will grow exponentially in the number of proof tree nodes. Clearly, the set of candidate properties must be restricted by some quality measure.

Property selection addresses two general issues. First, selecting prominent properties out of a set of possible properties can be seen as inducing a proper model that captures only the salient properties of the training data. This is one of the main tasks of statistical machine learning. Second, compact models will disallow overfitting the training data as could be done with models with one parameter per training element. Instead, a proper model will allow generalizations to new data and temper the overtraining problem.

Depending on the definition of properties (for the CLP application, e.g., as connected subgraphs of proof trees s.t. each node of a subgraph has either zero descendants or the same number of descendants as the corresponding node of the supergraph and the node sets of the subgraphs do not intersect) and the definition of a procedure to incrementally construct properties (e.g., by selecting from an initial set of query-node properties and from properties built by performing one-step resolutions at terminal nodes of subtree-properties of the model), we start from a set of candidate properties for a log-linear model.

But what should the above-mentioned quality measure be? We could take as the measure the improvement in log-likelihood when extending a model p_λ based upon weighted property function $\phi = \lambda \cdot \nu$ by a single candidate property[6] c with parameter α to a model $p_{\alpha+\lambda}$ based on extended property function $\alpha c + \phi$. In its basic form this quality measure would require a calculation of maximum likelihood estimates of extended models via the IM algorithm for each candidate property. Clearly, this is not feasible for models with large parameter spaces. Following Della Pietra, Della Pietra, and Lafferty (1995) or Berger, Della Pietra, and Della Pietra (1996), we could instead approximate the improvement due to adding a single property by adjusting only the parameter of this candidate and holding all other parameters of the model fixed. This would make the property selection algorithm practical but also greedy. Unfortunately, in constrast to this approach, we cannot directly maximize the gain of adding property c as a function of parameter α since the incomplete-data log-likelihood L is not concave in the parameters. However, we can define an auxiliary function similar to the one used in parameter estimation to express an approximate gain as a conservative estimate of the log-likelihood difference. A possible definition of an approximate gain can be derived from an instantiation of the auxiliary function A of Sect. 5.1 to $A(\alpha + \lambda)$, denoting the extension of a log-linear model p_λ with property-function vector ν by a single property c with log-parameter α.

Definition 7. *Let $\phi = \lambda \cdot \nu$ be a weighted property function, c be a candidate property, and $\alpha \in \mathbb{R}$ the log-parameter corresponding to c. Then the approximate gain $G_c(\alpha+\lambda)$ of adding candidate property c with parameter value α to the log-linear model p_λ is defined s.t.*

$$G_c(\alpha + \lambda) = \textstyle\sum_{y \in \mathcal{Y}} (1 + k_\lambda[\alpha c] - p_\lambda[e^{\alpha c}])$$

where $k_\lambda(x) = \frac{p_\lambda(x)}{\sum_{x \in X(y)} p_\lambda(x)}$, $p_\lambda(x) = Z_\phi^{-1} e^{\phi(x)} p_0(x)$.

For this function similar properties hold as for the auxiliary function A of Sect. 5.1. Since G_c is strictly concave in the parameters, we can maximize it directly and uniquely as a function of α.

Proposition 9. *For each $\lambda \in \Lambda$, $\alpha \in \mathbb{R}$: $G_c(\alpha + \lambda)$ takes its maximum as a function of α at the unique point $\hat{\alpha}$ satisfying*

$$\textstyle\sum_{y \in \mathcal{Y}} k_\lambda[c] = \sum_{y \in \mathcal{Y}} p_\lambda[c\, e^{\hat{\alpha} c}].$$

[6] In the following we will refer to the property corresponding to property function c as the "property c".

Proof. $\frac{\partial}{\partial\alpha} G_c(\alpha+\lambda) = \sum_{y\in\mathcal{Y}} (k_\lambda[c] - p_\lambda[c\, e^{\alpha c}])$,
$\frac{\partial^2}{\partial\alpha^2} G_c(\alpha+\lambda) = -\sum_{y\in\mathcal{Y}} p_\lambda[c^2 e^{\alpha c}] < 0$. □

Property selection then will incorporate that property out of the set of candidates that gives greatest improvement to the model at the property's best adjusted parameter value. Since we are interested only in relative, not absolute gains, a single, non-iterative maximization of the approximate gain will suffice to choose from the candidates.

Definition 8 (property selection)**.** *Let C be a set of candidate properties, $c \in C$ be a candidate property with log-parameter $\alpha \in \mathbb{R}$, and $G_c(\lambda) = max_\alpha G_c(\alpha+\lambda)$ the maximal approximate gain that property c can give to model p_λ. Then c is selected in a property selection step for model p_λ if $c = argmax_{c'\in C} G_{c'}(\lambda)$.*

5.3. **Summary.** The combined incomplete-data induction algorithm for log-linear models can be summarized as follows.

Input: *Initial model p_0, incomplete data set $\mathcal{Y}$.*

Output: *Log-linear model p^* on complete data set $\mathcal{X} = \bigcup_{y\in\mathcal{Y}} X(y)$ with selected property function vector ν^* and log-parameter vector $\lambda^* = argmax_{\lambda\in\Lambda} L(\lambda)$ where $\Lambda = \{\lambda | \; p_\lambda$ is a log-linear model on $\mathcal{X}$ based on p_0, ν^* and $\lambda \in \mathbb{R}^n\}$.*

Algorithm: 1. $p^{(0)} = p_0$,

2. *Property selection: For each candidate property $c \in C^{(n)}$, compute the gain $G_c(\lambda^{(n)}) = max_{\alpha\in\mathbb{R}} G_c(\alpha+\lambda^{(n)})$ and select property $\hat{c} = argmax_{c\in C^{(n)}} G_c(\lambda^{(n)})$.*
3. *Parameter estimation: Compute the maximum likelihood parameter value $\hat{\lambda} = argmax_{\lambda\in\Lambda} L(\lambda)$ where $\Lambda = \{\lambda | \; p_\lambda(x)$ is a log-linear distribution on $\mathcal{X}$ with initial model p_0, property function vector $\hat{\nu} = \nu^{(n)} \cup \hat{c}$, and $\lambda \in \mathbb{R}^n\}$.*
4. *Set $p^{(n+1)} = p_{\hat{\lambda}\cdot\hat{\nu}}$, $n = n+1$, go to 2.*

Returning to the sample program of Fig. 1, we can find a simple log-linear reformulation of the probabilistic CLP model as follows. In order to distinguish between the possible proof trees of Fig. 2 it is sufficient to define a single property referring to the variable binding either to a or to b. Taking a parameter value of $log\, 2$ for a single property involving the variable binding to a will yield the desired probability distribution $p(x_1) = 2/3$, $p(x_2) = 1/3$ and incomplete-data log-likelihood $L = .148$. The same result is obtained by taking a parameter value of $log\, 1/2$ for a single property involving the variable binding to b. All other properties will be unable to distinguish between proof trees x_1 and x_2 and thus give a uniform distribution over the proof trees and log-likelihood $L = .125$.

6. Approximation Methods

With the algorithms and proofs of the preceding section at hand, induction of log-linear models from incomplete data reduces to a calculation of expectations of simple functions. This calculation can be done by an explicit summation over the configuration space only for probabilistic processing models with a small, finite set of possible analyses. In case of large or infinite configuration spaces and complex parameter spaces these expectations can get intractable both analytically and numerically. Here approximation methods have to be used.

Following Della Pietra, Della Pietra, and Lafferty (1995) and Abney (1996), we can use a combination of the approximation techniques of Newton's method and Monte Carlo methods. In order to give a self-contained recipe for inducing log-linear models from incomplete data, we will make the proposed use of these methods explicit in the following.

Newton's method is a technique to approximate the solution α of an equation $f(\alpha) = 0$ by using a sequence of linearizations of f. At each step the intersection of the tangent to f at α_t with the α-axis is taken, yielding an improved estimate α_{t+1}. The iteration formulae to approach the solution up to a desired accuracy are defined as follows:

$$\alpha_{t+1} = \alpha_t - \frac{f(\alpha_t)}{f'(\alpha_t)} \textit{ where } f'(\alpha_t) \textit{ is the derivative of } f \textit{ at } \alpha_t.$$

This method directly suits our application when we replace $f(\alpha)$ by the first derivative of the auxiliary function A, $\frac{\partial}{\partial \gamma_i} A(\gamma + \lambda)$, in case of parameter estimation, and by the first derivative of the approximate gain G_c, $\frac{\partial}{\partial \alpha} G_c(\alpha + \lambda)$, in case of property selection. Newton's method usually converges rapidly for such functions.

The expectations expressed in the such defined Newton formulae then can be estimated by Monte Carlo methods. A Monte Carlo technique applicable to our problem is the Metropolis-Hastings method. The strategy behind this method is to generate a random sample from a target distribution p via choosing a nominating matrix p' from which sampling is easy and performing a Bernoulli trial with parameter α to determine whether to accept or reject the nominated sample point. That means, this method converts a sampler for p' into a sampler for p via the evaluation matrix α. For our application, we can take as nominating matrix for each query $y \in \mathcal{Y}$ a stochastic context-free CLP model $p(x; \pi)$ on $X(y)$ as defined in Sect. 3. From this stochastic derivation model sampling is easy and can be converted by a standard evaluation matrix to sampling from the desired log-linear distribution $p_\lambda(x)$ on $X(y)$. More formally, it can be shown that the distribution of the sampled random variables X_i will converge to the target distribution p_λ as $i \to \infty$, i.e., we have:

$$lim_{i\to\infty} P(X_i = x) = p_\lambda(x) \; for \; all \; x \in X(y).$$

Following standard textbooks such as Fishman (1996), an application of the Metropolis-Hastings algorithm to our problem is as follows.

Input: *initial state* $x_0 \in X(y)$,

nominating matrix $p' = p(x;\pi)$ *on* $X(y)$,

log-linear distribution $p = p_\lambda(x)$ *on* $X(y)$,

evaluation matrix $\alpha_{x,z} = \begin{cases} 1 & if \; p(x)p'(z) \leq p(z)p'(x) \\ \frac{p(z)p'(x)}{p(x)p'(z)} & if \; p(x)p'(z) > p(z)p'(x) \end{cases}$,

terminal number of steps k.

Output: *random sample* $X_0, \ldots, X_k$ *from* p_λ *on* $X(y)$.

Algorithm:

$X_0 := x_0$,
$i := 1$,
While $i \leq k$
  $x := X_{i-1}$,
  Randomly generate z *from* p',
  If $z = X_{i-1}$, *then* $X_i := X_{i-1}$,
    Else evaluate $\alpha_{x,z}$,
    Randomly generate u *from uniform distribution on* $[0,1]$,
    If $u \leq \alpha_{x,z}$, *then* $X_i := z$,
      Else $X_i := X_{i-1}$,
  $i := i + 1$,
return $X_0, \ldots, X_k$.

In general, a proper random sample from a probability distribution p allows the estimation of expectations of functions f with respect to p directly from the sample points X_i, i.e., we have:

$$lim_{K\to\infty} \frac{1}{K} \sum_{i=1}^{K} f(X_i) = \sum_x f(x)p(x).$$

For our application taking a random sample $\tilde{X}(y)$ from p_λ on $X(y)$ for a query $y \in \mathcal{Y}$ will allow us to calculate expectations of functions with respect to the distribution p_λ on $X(y)$. A combination of i-ary random samples $\tilde{X}(y)$ from p_λ on $X(y)$ for queries $y \in \mathcal{Y}$ will yield a combined random sample $\tilde{\mathcal{X}} = \bigcup_{y\in\mathcal{Y}} \tilde{X}(y)$ from p_λ on $\mathcal{X} = \bigcup_{y\in\mathcal{Y}} X(y)$. From this random sample we can then estimate expectations of functions with respect to a distribution of p_λ on $\mathcal{X}$.

Returning to the estimation of the expectations involved in our induction formulae, we note that we can use the same random sample from $p^{(n)}$ for each iteration of Newton's method in estimating the gain $G_c(\lambda^{(n)})$ for each candidate property $c \in C^{(n)}$ simultaneously. After adding a selected property $\hat{c}$ to the model, we can again use a single random sample from the extended model for the estimation of the maximum likelihood parameter values via Newton's method for each property in parallel. This means that we can build up hash-tables counting up how many times each property takes on which value. Let $\mathcal{Y}$ be an incomplete data sample of size N, $\tilde{X}(y)$ be a complete data sample of size M for y, and $\tilde{\mathcal{X}}$ be a combined complete data sample of size L. Then the relevant hash-tables can be defined as follows:

1. $S_{c,v} = \sum_{\tilde{x} \in \tilde{\mathcal{X}}} [\![c(\tilde{x}) = v]\!]$ *is the number of times property function* c *takes value* v *in combined random sample* $\tilde{\mathcal{X}}$,
2. $T_{y,c,v} = \sum_{\tilde{x} \in \tilde{X}(y)} [\![c(\tilde{x}) = v]\!]$ *is the number of times property function* c *takes value* v *in random sample* $\tilde{X}(y)$,
3. $U_{i,m} = \sum_{\tilde{x} \in \tilde{\mathcal{X}} |\ \nu_{\#}(\tilde{x}) = m} \nu_i(\tilde{x})$ *is the number of times property* χ_i *appears in combined random sample* $\tilde{\mathcal{X}}$ *when there is a total number of* m *property instances for each sample point.*

Furthermore, it will be convenient to define the following variables:

$$s_r(\alpha, c) = \textstyle\sum_v S_{c,v} e^{\alpha v} v^r,$$
$$t_y(c) = \textstyle\frac{1}{M} \sum_v T_{y,c,v} v,$$
$$u_r(\alpha, i) = \textstyle\sum_m U_{i,m} e^{\alpha m} m^r.$$

The expectations involved in Newton's formulae for the property selection task can then be approximated by random sample counts as follows:

$$\begin{aligned}
\alpha_{t+1} &= \alpha_t + \frac{\frac{\partial}{\partial \alpha_t} G_c(\alpha_t + \lambda)}{\frac{\partial^2}{\partial \alpha_t^2} G_c(\alpha_t + \lambda)} \\
&= \alpha_t + \frac{\sum_y k_\lambda[c] - N p_\lambda[c\, e^{\alpha_t c}]}{N p_\lambda[c^2 e^{\alpha_t c}]} \\
&\approx \alpha_t + \frac{\sum_y t_y(c) - \frac{N}{L} s_1(\alpha_t, c)}{\frac{N}{L} s_2(\alpha_t, c)}.
\end{aligned}$$

Similar estimation formulae can be obtained for the task of parameter estimation:

$$
\begin{aligned}
\alpha_{t+1} &= \alpha_t + \frac{\frac{\partial}{\partial \alpha_t} A(\gamma + \lambda)}{\frac{\partial^2}{\partial \alpha_t^2} A(\gamma + \lambda)} \\
&= \alpha_t + \frac{\sum_y k_\lambda[\nu_i] - N p_\lambda[\nu_i e^{\alpha_t \nu_\#}]}{N p_\lambda[\nu_i \nu_\# e^{\alpha_t \nu_\#}]} \\
&\approx \alpha_t + \frac{\sum_y t_y(\nu_i) - \frac{N}{L} u_0(\alpha_t, i)}{\frac{N}{L} u_1(\alpha_t, i)}.
\end{aligned}
$$

7. Search Methods

The induction and approximation techniques of the preceding sections provide the means to induce a proper probability distribution over analyses of a log-linear probabilistic processing model from unanalyzed input data. In case of ambiguity, this allows us to distinguish between analyses according to a well-defined and practical quality measure. However, if we are interested only in the best analysis of a given input, so far a ranking of analyses requires a listing of analyses in order to choose the best one. Clearly, it would be nice to have search techniques like Viterbi's algorithm (Viterbi 1967), which works well for probabilisitic processing models based on context-free stochastic derivation processes.

Viterbi's algorithm is built upon a table of derivation states, called a chart, describing different pending derivations. During derivation, each state must keep track of the most probable path of states leading towards it. When the final state is reached, the maximum probability derivation can be recovered by tracing back the path of the best predecessor states. Different specifications of the algorithm depend on the chosen parsing strategy and the underlying probabilistic model.

In the following we will sketch one possibility to transfer these ideas to a method of probabilistic parsing in the area of CLP. For this aim we rely on the well-known parsing algorithm of Earley deduction. This technique provides the necessary chart structure accompanied with a simple parsing strategy. Depending on the specific definition of the property vector in the underlying log-linear CLP model, different definitions of the propagation of probabilities during the parsing process are possible. Since the property vector is considered to be an open parameter in our setting, we will not present a definitive solution to this problem but only give some rules of thumb how to proceed for some general examples.

Earley deduction was introduced by Pereira and Warren (1983) as a generalization of Earley's context-free parsing algorithm (see Aho and Ullman (1972)) to a parsing algorithm for definite clause grammars. Extensions of this method in the general setting of the CLP scheme of Höhfeld and Smolka (1988) have been presented, e.g., by Dörre (1993) and Dörre and Johnson (1995). The basic concepts of Earley deduction for CLP can be described as follows: Earley deduction works on two sets of definite clauses, the set of program clauses $\mathcal{P}$ and the set of derived clauses constituting the chart $\mathcal{C}$. An active item corresponds to a definite clause with at least one relational atom on its righthandside, i.e., to a non-unit clause. Passive items correspond to clauses whose righthandsides consist only of an $\mathcal{L}$-constraint, i.e., to unit-clauses. The input to the algorithm consists of a set of program clauses $\mathcal{P}$ and a query G. The content of the chart $\mathcal{C}$ initially consists of G and is continually added to by the following inference rules:

Prediction:

$$\frac{\begin{array}{l} c_1 = (H_1 \leftarrow B_1) \ \in \mathcal{C} \\ c_2 = (H_2 \leftarrow B_2) \ \in \mathcal{P} \end{array}}{c_3 = (C \leftarrow B_2' \cup \phi) \ \in \mathcal{C}}$$

where c_1 is non-unit, c_2 is unit or non-unit, C is the selected literal in B_1, ϕ is the $\mathcal{L}$-constraint in B_1, and there exists a variant $c_2' = (C \leftarrow B_2')$ of c_2 s.t. $\mathsf{V}(c_1) \cap \mathsf{V}(B_2') \subseteq \mathsf{V}(C)$.

Completion:

$$\frac{\begin{array}{l} c_1 = (H_1 \leftarrow B_1) \ \in \mathcal{C} \\ c_2 = (H_2 \leftarrow B_2) \ \in \mathcal{C} \end{array}}{c_3 = (H_1 \leftarrow (B_1 \setminus C) \cup B_2') \ \in \mathcal{C}}$$

where c_1 is non-unit, c_2 is unit, C is the selected literal in B_1, and there exists a variant $c_2' = (C \leftarrow B_2')$ of c_2 s.t. $\mathsf{V}(c_1) \cap \mathsf{V}(B_2') \subseteq \mathsf{V}(C)$.

A probabilistic version of a context-free Earley parser was presented in Stolcke (1993). In this framework, during derivation each completed state keeps track of the most probable path of states contributing to it. The probability propagation is done recursively by associating each predicted state with the probability of the corresponding rule and taking at each completion step the maximum of all products of probabilities of two states from which the completed state is derivable. When the final state is reached, the most probable analysis easily can be retrieved by building up a tree in accordance with the most probable path of states leading to the final completion.

If the property vector of a log-linear CLP model is defined s.t. properties are identified with program clauses, then the above model can be used also for probabilistic Earley deduction: During deduction, each predicted clause is associated with a weight corresponding to the clause-property used in the prediction. For each completed clause, the pair of clauses contributing with maximal product of weights to the completion is recorded. Given a procedure to construct a proof tree from a sequence of clauses linked by prediction and completion, the highest weighted partial proof tree corresponding to a completed clause can be constructed recursively and uniquely from the highest weighted pair of clauses contributing to the completion.

Unfortunately, weight propagation will get more complicated as we allow more complicated properties in our underlying log-linear CLP model. In case properties are identified with program clauses, completion means complete reduction of selected atoms using appropriate clauses. A numerical comparison between different ways of arriving at the same completed state can be done at every completion step. In contrast to this, if properties are allowed to be subtrees of proof trees, completion means completely building up a subtree of a proof tree during derivation. A numerical comparison between to ways of "completing" the same subtree in the same completion state might have to wait for several completion steps until the subtree is completely built up. Considering the possibility of a backward construction of the most probable proof tree in this setting, we cannot rely on an easy recording of the most probable path of clauses leading to the final completion state. Instead, in order to compare between the weights of the partial derivations contributing to such a "subtree-completion", we have to incrementally build up partial proof trees and check their properties during derivation.

Let subtree-properties be defined as follows: A subtree of a proof tree is a connected subgraph of a proof tree, each node of a subgraph has either zero descendants or the same number of descendants as the corresponding node of the supergraph, and the node sets of every two subtrees in the set of properties do not intersect. Then a simple recursive procedure to build up partial proof trees from completed states can be defined as follows:

For each completed state c_k, for each pair of states c_i, c_j from which c_k is derivable by completion, the partial proof tree t_{ij} corresponding to the completion of state c_k from states c_i, c_j is constructed s.t. $t_{ij} =$

1. $\begin{array}{c} t_i \\ \oplus \\ t_j \end{array}$ *, if c_i, c_j are completed states with trees t_i, t_j,*

   and $\begin{array}{c} t_1 \\ \oplus \\ t_2 \end{array} = \begin{array}{c} \vdots \\ A \\ | \\ B \cup C \\ | \\ D \\ \vdots \end{array}$ *if* $t_1 = \begin{array}{c} \vdots \\ A \\ | \\ B \end{array}$, $t_2 = \begin{array}{c} C \\ | \\ D \\ \vdots \end{array}$,

2. $\begin{array}{c} t_i \\ \otimes \\ t_j \end{array}$ *, if c_i is a predicted state $(E \leftarrow F)$ with tree* $t_i = \begin{array}{c} E \\ | \\ F \end{array}$,

   c_j is a completed state with tree t_j,

   and $\begin{array}{c} t_1 \\ \otimes \\ t_2 \end{array} = \begin{array}{c} \vdots \\ A \\ | \\ B \\ | \\ B \setminus C \cup D \\ \vdots \end{array}$ *if* $t_1 = \begin{array}{c} \vdots \\ A \\ | \\ B \end{array}$, $t_2 = \begin{array}{c} C \\ | \\ D \\ \vdots \end{array}$,

3. $\begin{array}{c} t_i \\ | \\ t_j \end{array}$ *, if c_i is a predicted state $(E \leftarrow F)$ with tree* $t_i = \begin{array}{c} E \\ | \\ F \end{array}$,

   c_j is a predicted state including $\mathcal{L}$-constraint ϕ and with tree $t_j = \phi$.

During derivation, for each "property-completion" at some completed state c_k, the variable t_k denoting the partial proof tree corresponding to c_k is instantiated to the most probable partial proof tree t_{ij} which can be built from all states c_i, c_j contributing to the completion of c_k:

Let p_λ be a log-linear distribution on the set $\mathcal{X}$ of proof trees of a constraint logic program $\mathcal{P}$ with property vector χ and property function vector ν. Then for each completed state c_k, for each property $\chi_n \in \chi$, for each partial proof tree t_{ij} constructable for c_k from trees t_i, t_j s.t. $\nu_n(t_{ij}) > \nu_n(t_i) + \nu_n(t_j)$, set $t_k = argmax_{t_{ij}}\, p_\lambda(t_{ij})$.

For the above definition of subtree-properties, this procedure guarantees that the most probable proof tree is built up during derivation. The possible savings in computational complexity induced by this procedure clearly depend on the size of the subtree-properties to be worked out during derivation. However, if subtree-properties are allowed to be overlapping or disconnected subgraphs of proof trees, then the above dynamic programming approach is no longer applicable. In this case either exhaustive search or approximation methods are required.

8. Conclusion

We presented a log-linear probability model for probabilistic CLP. On top of this model we defined an algorithm to estimate the parameters and to select the properties of log-linear models from incomplete data. This algorithm is an extension of the iterative scaling algorithm of Della Pietra, Della Pietra, and Lafferty (1995) adjusted to incomplete data. The algorithm applies to log-linear models in general and is accompanied with suitable approximation methods when applied to large data spaces. Furthermore, we presented an approach to search for most probable analyses of the probabilistic CLP model. This can be useful for the ambiguity resolution problem in natural language processing applications.

Compared with Abney's approach to a log-linear model for stochastic attribute-value grammars, our approach adds the important aspect of incomplete data to the parameter estimation and property selection problem. Furthermore, we investigate the problem of searching for best analyses which is not addressed by Abney (1996).

The expressive power of log-linear models even allows us to couch other approaches to probabilisitic processing beyond context-freeness in terms of this framework. Statistical decision trees as used in the probabilistic parsing model of Magerman (1994) can be cast in the log-linear framework by encoding the questions building up a decision tree as binary-valued, disjoint property functions. Property selection then can be seen as closely related to growing a decision tree and iterative maximization can be seen as maximum likelihood estimation for such defined decision trees. However, in contrast to the algorithms used by Magerman (1994), which require large samples of complete data, our approach allows induction of the probabilistic model from incomplete data.

A similar statement can be made for the probabilistic tree substitution model of Bod (1995). This approach can be couched as a log-linear model employing all subtrees of a tree bank, which is annotated according to some grammar framework, as properties of the model. Again, Bod's approach relies on hand-analyzed data and does not allow to estimate the probabilistic model from unanalyzed input. Furthermore, this approach does not provide a means to automatically select subtree-properties from the exponentially many candidates.

Clearly, our model of probabilistic CLP is not the last word on probabilistic processing beyond context-freeness. As mentioned above, log-linear models are closely related to other probabilistic models such as random fields (Geman 1990), graphical networks (Pearl 1988) or neural networks (Ackley and Hinton 1985). Future work should exploit this resemblance in order to learn from related techniques to induce, approximate or search in log-linear probability models. Furthermore, the possibilities of our powerful processing model shall be applied to natural language processing tasks other than ambiguity resolution.

References

Abney, S. (1996). Stochastic attribute-value grammars. Unpublished manuscript, University of Tübingen. To appear in Computational Linguistics.

Ackley, D. H. and G. E. Hinton (1985). A learning algorithm for Boltzmann machines. *Cognitive Science 9*, 147–169.

Aho, A. V. and J. D. Ullman (1972). *The Theory of Parsing, Translation and Compiling*, Volume I: Parsing. NJ: Prentice-Hall.

Baker, J. (1979). Trainable grammars for speech recognition. In D. Klatt and J. Wolf (Eds.), *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pp. 547–550.

Baum, L. E. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities III*, 1–8.

Baum, L. E. and J. A. Eagon (1967). An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the American Mathematical Society 73*(1), 360–363.

Baum, L. E., T. Petrie, G. Soules, and N. Weiss (1970). A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics 41*(1), 164–171.

Beeferman, D., A. Berger, and J. Lafferty (1997a). A model of lexical attraction and repulsion. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, Madrid, Spain.

Beeferman, D., A. Berger, and J. Lafferty (1997b). Text segmentation using exponential models. In *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing.*

Berger, A. L., V. J. Della Pietra, and S. A. Della Pietra (1996). A maximum entropy approach to natural language processing. *Computational Linguistics 22*(1), 39–71.

Bod, R. (1995). *Enriching Linguistics with Statistics: Performance Models of Natural Language*. Ph. D. thesis, Institute for Logic, Language and Computation, Universiteit van Amsterdam.

Briscoe, T. and N. Waegner (1992). Robust stochastic parsing using the inside-outside algorithm. In *Proceedings of AAAI92 Workshop on Probabilistically-Based Natural Language Processing Techniques*, San Jose, CA.

Carroll, G. and E. Charniak (1992). Two experiments on learning probabilistic dependency grammars from corpora. Technical Report RI 02912, Department of Computer Science, Brown University, Providence RI.

Darroch, J. and D. Ratcliff (1972). Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics 43*(5), 1470–1480.

Della Pietra, S., V. Della Pietra, and J. Lafferty (1995). Inducing features of random fields. Technical Report CMU-CS-95-144, CMU. Also appeared in IEEE Transactions on Pattern Analysis and Machine Intelligence 19(4), pp. 380-393, April 1997.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society 39*(B), 1–38.

Dörre, J. (1993). Generalizing Earley deduction for constraint-based grammars. In J. Dörre (Ed.), *Computational Aspects of Constraint-Based Linguistic Description I*, pp. 25–41. DYANA-2 Deliverable R1.2.A.

Dörre, J. and M. Dorna (1993). CUF - a formalism for linguistic knowledge representation. In J. Dörre (Ed.), *Computational Aspects of Constraint-Based Linguistic Description I*, pp. 3–22. DYANA-2 Deliverable R1.2.A.

Dörre, J. and M. Johnson (1995). Memoization of coroutined constraints. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pp. 100–107.

Eisele, A. (1994). Towards probabilistic extensions of constraint-based grammars. In J. Dörre (Ed.), *Computational Aspects of Constraint-Based Linguistic Description II*, pp. 3–21. DYANA-2 Deliverable R1.2.B.

Fishman, G. S. (1996). *Monte Carlo. Concepts, Algorithms and Applications.* Berlin: Springer.

Geman, D. (1990). Random fields and inverse problems in imaging. In P. L. Hennequin (Ed.), *École d'Été de Probabilités de Saint-Flour XVIII - 1988*, pp. 117–193. Berlin: Springer. Lecture Notes in Mathematics, 1427.

Geman, S. and D. Geman (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-6*, 721–741.

Götz, T. (1995). Compiling HPSG constraint grammars into logic programs. In *Workshop on Computational Logic and Natural Language Processing*, Edinburgh.

Höhfeld, M. and G. Smolka (1988). Definite relations over constraint languages. LILOG Report 53, IBM Deutschland, Stuttgart.

Jaffar, J. and J.-L. Lassez (1986). Constraint logic programming. Technical report 74, Department of Computer Science, Monash University.

Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical Review 106*, 620–630.

Jaynes, E. T. (1983). *Papers on Probability, Statistics and Statistical Physics.* Dortrecht: D. Reidel. ed. by R. D. Rosenkrantz.

Jelinek, F., J. D. Lafferty, and R. L. Mercer (1990). Basic methods of probabilistic context free grammars. Technical report, Continuous Speech Recognition Group IBM - T.J. Watson Research Center, Yorktown Heights, NY.

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

Lloyd, J. W. (1987). *Foundations of Logic Programming.* Berlin: Springer.

Magerman, D. M. (1994). *Natural Language Parsing as Statistical Pattern Recognition.* Ph. D. thesis, Department of Computer Science, Stanford University.

Mark, K., M. Miller, U. Grenander, and S. Abney (1992). Parameter estimation for constrained context-free language models. In *DARPA Speech and Natural Language Workshop*, Harriman, New York.

Miyata, T. (1996). *A Study on Inference Control in Natural Language Processing.* Ph. D. thesis, Graduate School of the University of Tokyo, Tokyo, Japan.

Osborne, M. and T. Briscoe (1997). Learning stochastic categorial grammars. In *CoNLL97: Proceedings of the Workshop on Computational Natural Language Learning*, Madrid, Spain, pp. 80–87.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. CA: Morgan Kaufmann.

Pereira, F. and Y. Schabes (1992). Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, Newark, Delaware, pp. 128–135.

Pereira, F. C. N. and D. H. D. Warren (1983). Parsing as deduction. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, Boston, MA, pp. 137–144.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE, Vol. 77, No. 2*, pp. 257–285.

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing*.

Resnik, P. (1992). Probabilistic tree-adjoining grammars as a framework for statistical natural language processing. In *Proceedings of COLING-92*, Nantes, pp. 418–424.

Rosenfeld, R. (1996). A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language 10*, 187–228.

Schabes, Y. (1992). Stochastic lexicalized tree-adjoining grammars. In *Proceedings of COLING-92*, Nantes, pp. 426–432.

Stolcke, A. (1993). An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. Technical Report TR-93-065, International Computer Science Institute, Berkeley, CA.

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

Graduiertenkolleg ILS, Seminar für Sprachwissenschaft, Universität Tübingen, Wilhelmstr. 113, 72074 Tübingen, Germany

E-mail address: `riezler@sfs.nphil.uni-tuebingen.de`