Towards Multistrategic Statistical Relational Learning

Marenglen Biba, Stefano Ferilli, Floriana Esposito

Abstract Statistical Relational Learning (SRL) is a growing field in Machine Learning that aims at the integration of logic-based learning approaches with probabilistic graphical models. Markov Logic Networks (MLNs) are one of the state-of-the-art SRL models that combine first-order logic and Markov networks (MNs) by attaching weights to first-order formulas and viewing these as templates for features of MNs. Learning models in SRL consists in learning the structure (logical clauses in MLNs) and the parameters (weights for each clause in MLNs). Structure learning of MLNs is performed by maximizing a likelihood function (or a function thereof) over relational databases and MLNs have been successfully applied to problems in relational and uncertain domains. However, most complex domains are characterized by incomplete data. Until now SRL models have mostly used Expectation-Maximization (EM) for learning statistical parameters under missing values. Multistrategic learning in the relational setting has been a successful approach to dealing with complex problems where multiple inference mechanisms can help solve different subproblems. Abduction is an inference strategy that has been proven useful for completing missing values in observations. In this paper we propose two frameworks for integrating abduction in SRL models. The first tightly integrates logical abduction with structure and parameter learning of MLNs in a single step. During structure search guided by conditional likelihood, clause evaluation is performed by first trying to logically abduce missing values in the data and then by learning optimal pseudo-likelihood parameters using the completed data. The second approach integrates abduction with Structural EM of [17] by performing logical abductive inference in the E-step and then by trying to maximize parameters in the M-step.

Marenglen Biba e-mail: biba@di.uniba.it
· Stefano Ferilli e-mail: ferilli@di.uniba.it
· Floriana Esposito e-mail: esposito@di.uniba.it

Department of Computer Science, University of Bari, Via E. Orabona 4, 70124, Italy

1 Introduction

Traditionally, Machine Learning research has fallen into two separate subfields: one that has focused on logical representations, and one on statistical ones. Logical Machine Learning approaches based on logic programming, description logics, classical planning, rule induction, etc, tend to emphasize handling complexity. Statistical Machine Learning approaches like Bayesian networks, hidden Markov models, statistical parsing, neural networks, etc, tend to emphasize handling uncertainty. However, learning systems must be able to handle both for real-world applications. The first attempts to integrate logic and probability were made in Artifical Intelligence and date back to the works in [2, 22, 46]. Later, several authors began using logic programs to compactly specify Bayesian networks, an approach known as knowledge-based model construction [68].

A central problem in Machine Learning has always been learning in rich representations that enable to deal with structure and relations. Much progress has been achieved in the relational learning field or differently known as Inductive Logic Programming [33]. On the other hand, successful statistical machine learning models with their roots in statistics and pattern recognition, have made possible to deal with noisy and uncertain domains in a robust manner. Powerful models such as Probabilistic Graphical Models [48] and related algorithms have the power to handle uncertainty but lack the capability of dealing with structured domains.

In Machine Learning, recently, in the burgeoning field of Statistical Relational Learning (SRL) [20] or Probabilistic Inductive Logic Programming [11], several approaches for combining logic and probability have been proposed. A growing amount of work has been dedicated to integrating subsets of first-order logic with probabilistic graphical models, to extend logic programs with a probabilistic semantics or integrate other formalisms with probability. Some of the logic-based approaches are: Knowledge-based Model Contruction [68], Bayesian Logic Programs [28], Stochastic Logic Programs [41, 9], Probabilistic Horn Abduction [51], Queries for Probabilistic Knowledge Bases [44], PRISM [60], CLP(BN) [59]. Other approaches include frame-based systems such as Probabilistic Relational Models [43] or PRMs extensions defined in [47], description logics based approaches such as those in [8] and P-CLASSIC of [30], database query langauges [67], [54], etc.

All these approaches combine probabilistic graphical models with subsets of first-order logic (e.g., Horn Clauses). One of the state-of-the-art SRL approches is Markov logic [58], a powerful representation that has finite first-order logic and probabilistic graphical models as special cases. It extends first-order logic by attaching weights to formulas providing the full expressiveness of graphical models and first-order logic in finite domains and remaining well defined in many infinite domains [58, 65]. Weighted formulas are viewed as templates for constructing Markov Networks (MNs) and in the infinite-weight limit, Markov logic reduces to standard first-order logic. In Markov logic it is avoided the assumption of i.i.d. (independent and identically distributed) data made by most statistical learners by using the power of first-order logic to compactly represent dependencies among objects and relations. In this paper we will focus on this SRL model.

The representation power and the robustness of SRL models to deal with uncertainty does not solve all the problems present in complex domains. Dealing with unknown or partially observed data is an important problem in Machine Learning. Most SRL models face this problem only from the parameter setting point of view by following similar approaches developed in the statistical machine learning field. The most used approach is Expectation-Maximization (EM) [13]. On the other side, in relational learning different approaches have been proposed that integrate multiple inference mechanisms in inductive learning to deal with incomplete data [26, 16].

Multistrategic approaches to Machine Learning [38] aim at combining different inference strategies in order to take advantage of each of these during learning. One of these inference mechanisms is abduction. In the general inference schema, the fundamental equation $BK \cup T \models O$ involves a language L, a background knowledge BK and a theory T, that contains concept definitions accounting for some observations O. Specifically, O stands for the extensional representation of concepts, while T is an intensional description, expressed in L, that explains such concepts together with BK. Deduction traces forward the equation, deriving O given T and BK , and hence it is a truth-preserving inference. Conversely, tracing the equation backward yields two falsity-preserving inferences (meaning that if O is false, then the hypothesis cannot be true): induction, when T is to be hypothesized given O and BK , or abduction, when BK is to be hypothesized given O and T (i.e., plausible/likely causes of given observations). Most approaches to relational learning rely on inductive mechanisms to fine-tune T in order to achieve the learning goal, but problems might arise due to the partial relevance of the available evidence O. Abduction could be exploited to overcome such a limitation by bridging the observations relevance gap. Indeed, it is able to capture default reasoning [57], a well-known form of reasoning to deal with incomplete information [27, 50]. Thus, making these inference strategies work together would allow to take advantage of the benefits that each of them can bring. A step in this direction was proposed in [26], where the authors show how to learn with incomplete background data about the training examples by exploiting the hypothetical reasoning of abduction. Another approach is that in [16] where it was proposed a framework for the integration of abductive and inductive learning in an incremental ILP system.

In this paper we propose two frameworks that integrate logical abduction in an SRL model based on Markov logic. The novelty of the proposed approaches stands in the tight integration of structure and parameter learning of an SRL model in a single step inside which a logical abductive proof procedure and a statistical parameter estimation method are exploited. The first framework integrates logical abduction with structure and parameter learning of MLNs in a single step. During structure search guided by conditional likelihood, structure evaluation is performed by first trying to logically abduce missing values in the data and then by learning optimal pseudo-likelihood parameters using the completed data. The second approach integrates abduction with Structural EM of [17] by performing logical abductive inference in the E-step and then by trying to maximize parameters in the M-step.

2 Markov Networks and Markov Logic Networks

A Markov network (also known as Markov random field) is a model for the joint distribution of a set of variables X = (X_1,X_2,...,X_n) $\in \chi$ [12]. It is composed of an undirected graph G and a set of potential functions. The graph has a node for each variable, and the model has a potential function ϕ_k for each clique in the graph. A potential function is a non-negative real-valued function of the state of the corresponding clique. The joint distribution represented by a Markov network is given by:

$$P(X = x) = \frac{1}{Z}\prod_k \phi_k(x_{\{k\}}) \tag{1}$$

where $x_{\{k\}}$ is the state of the *k*th clique (i.e., the state of the variables that appear in that clique). *Z*, known as the partition function, is given by:

$$Z = \sum_{x\in\chi}\prod_k \phi_k(x_{\{k\}}) \tag{2}$$

Markov networks are often conveniently represented as log-linear models, with each clique potential replaced by an exponentiated weighted sum of features of the state, leading to:

$$P(X = x) = \frac{1}{Z}\exp(\sum_j w_j f_j(x)) \tag{3}$$

A feature may be any real-valued function of the state. We will focus on binary features,$f_j \in \{0,1\}$. In the most direct translation from the potential-function form, there is one feature corresponding to each possible state x_k of each clique, with its weight being $\log(\phi(x_{\{k\}})$. This representation is exponential in the size of the cliques. However a much smaller number of features (e.g., logical functions of the state of the clique) can be specified, allowing for a more compact representation than the potential-function form, particularly when large cliques are present. MLNs take advantage of this.

A first-order KB can be seen as a set of hard constraints on the set of possible worlds: if a world violates even one formula, it has zero probability. The basic idea in Markov logic is to soften these constraints: when a world violates one formula in the KB it is less probable, but not impossible. The fewer formulas a world violates, the more probable it is. Each formula has an associated weight that reflects how strong a constraint it is: the higher the weight, the greater the difference in log probability between a world that satisfies the formula and one that does not, other things being equal.

A Markov logic network [58] *L* is a set of pairs $(F_i; w_i)$, where F_i is a formula in first-order logic and w_i is a real number. Together with a finite set of constants $C = \{c_1, c_2, \ldots, c_p\}$ it defines a Markov network $M_{L;C}$ as follows:

1. $M_{L;C}$ contains one binary node for each possible grounding of each predicate appearing in L. The value of the node is 1 if the ground predicate is true, and 0 otherwise.

2. $M_{L;C}$ contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the w_i associated with F_i in L. Thus there is an edge between two nodes of $M_{L;C}$ iff the corresponding ground predicates appear together in at least one grounding of one formula in L. An MLN can be viewed as a template for constructing Markov networks. The probability distribution over possible worlds x specified by the ground Markov network $M_{L;C}$ is given by

$$P(X=x) = \frac{1}{Z}\exp(\sum_{i=1}^{F} w_i n_i(x)) = \frac{1}{Z}\prod_i \phi_i(x_i)^{n_i(x)} \tag{4}$$

where F is the number of formulas in the MLN and $n_i(x)$ is the number of true groundings of F_i in x. As formula weights increase, an MLN increasingly resembles a purely logical KB, becoming equivalent to one in the limit of all infinite weights.

In this paper we focus on MLNs whose formulas are function-free clauses and assume domain closure (it has been proven that no expressiveness is lost), ensuring that the Markov networks generated are finite. In this case, the groundings of a formula are formed simply by replacing its variables with constants in all possible ways.

3 Structure and Parameter Learning of MLNs

3.1 Generative Structure Learning of MLNs

One of the approaches for learning MN weights is iterative scaling [12]. However, maximizing the likelihood (or posterior) using a quasi-Newton optimization method like L-BFGS has recently been found to be much faster [62]. Regarding structure learning, the authors in [12] induce conjunctive features by starting with a set of atomic features (the original variables), conjoining each current feature with each atomic feature, adding to the network the conjunction that most increases likelihood, and repeating. The work in [37] extends this to the case of conditional random fields, which are Markov networks trained to maximize the conditional likelihood of a set of outputs given a set of inputs.

The first attempt to learn MLNs was that in [58], where the authors used CLAUDIEN [10] to learn the clauses of MLNs and then learned the weights by maximizing pseudo-likelihood. In [29] another method was proposed that combines ideas from ILP and feature induction of Markov networks. This algorithm, that performs a beam or shortest first search in the space of clauses guided by a weighted pseudo-log-likelihood (WPLL) measure [3], outperformed that of [58]. Recently, in [39] a bottom-up approach was proposed in order to reduce the search space. This algorithm uses a propositional Markov network learning method to construct template networks that guide the construction of candidate clauses. In this way, it generates fewer candidates for evaluation. In [5], the authors proposed an algorithm based on the iterated local search metaheuristic and showed that using parallel computation, it is possible to improve over the previous algorithms. For every candidate structure, in all these algorithms, the parameters that optimize the WPLL are set through L-BFGS that approximates the second-derivative of the WPLL by keeping a running finite-sized window of previous first-derivatives.

3.2 Discriminative Structure and Parameter Learning of MLNs

Learning MLNs in a discriminative fashion has produced for predictive tasks much better results than generative approaches as the results in [64] show. In this work the voted-perceptron algorithm was generalized to arbitrary MLNs by replacing the Viterbi algorithm with a weighted satisfiability solver. The new algorithm is essentially gradient descent with an MPE approximation to the expected sufficient statistics (true clause counts) and these can vary widely between clauses, causing the learning problem to be highly ill-conditioned, and making gradient descent very slow. In [36] a preconditioned scaled conjugate gradient approach is shown to outperform the algorithm in [64] in terms of learning time and prediction accuracy. This algorithm is based on the scaled conjugate gradient method and very good results are obtained with a simple approach: per-weight learning weights, with the weight's learning rate being the global one divided by the corresponding clause's empirical number of true groundings.

However, for both these algorithms the structure is supposed to be given by an expert or learned previously and they focus only on the parameter learning task. This can lead to suboptimal results if the clauses given by an expert do not capture the essential dependencies in the domain in order to improve classification accuracy. On the other side, since to the best of our knowledge, no attempt has been made to learn the structure of MLNs discriminatively, the clauses learned by generative structure learning algorithms tend to optimize the joint distribution of all the variables and applying discriminative weight learning after the structure has been learned generatively may lead to suboptimal results since the initial goal of the learned structure was not to discriminate query predicates.

Recently different attempts have been proposed for discriminative structure learning of MLNs. In [24] MLNs were restricted to non recursive definite clauses and the ILP system ALEPH [66] was used to generate a large number of potentially good candidates that are then scored using exact inference methods. In [4] the authors proposed another approach, they set parameters by maximizing likelihood and choose structures by conditional likelihood. Inference for each canidate clause is performed using the lazy version of the MC-SAT algorithm [53]. The authors propose some simple heuristics to make the problem tractable and show improvements in terms of predictive accuracy over generative structure learning approaches and discriminative weight learning algorithms.

4 First-Order Logic and Inductive Logic Programming

Relational learning is mostly related to first-order logic or more restricted formalisms. A first-order knowledge base (KB) is a set of sentences or formulas in first-order logic (FOL) [19]. Formulas in FOL are constructed using four types of symbols: constants, variables, functions, and predicates. Constant symbols represent objects in the domain of interest. Variable symbols range over the objects in the domain. Function symbols represent mappings from tuples of objects to objects. Predicate symbols represent relations among objects in the domain or attributes of objects. A term is any expression representing an object in the domain. It can be a constant, a variable, or a function applied to a tuple of terms. An atomic formula or atom is a predicate symbol applied to a tuple of terms. A ground term is a term containing no variables. A ground atom or ground predicate is an atomic formula all of whose arguments are ground terms. Formulas are recursively constructed from atomic formulas using logical connectives and quantifiers. A positive literal is an atomic formula; a negative literal is a negated atomic formula. A KB in clausal form is a conjunction of clauses, a clause being a disjunction of literals. A definite clause is a clause with exactly one positive literal (the head, with the negative literals constituting the body). A possible world or Herbrand interpretation assigns a truth value to each possible ground predicate.

Because of the computational complexity, KBs are generally constructed using a restricted subset of FOL where inference and learning is more tractable. The most widely-used restriction is to Horn clauses, which are clauses containing at most one positive literal. In other words, a Horn clause is an implication with all positive antecedents, and only one (positive) literal in the consequent. A program in the Prolog language is a set of Horn clauses. Prolog programs can be learned from examples (often relational databases) by searching for Horn clauses that hold in the data. The field of inductive logic programming (ILP) [33] deals exactly with this problem. The main task in ILP is finding an hypothesis H (a logic program, i.e. a definite clause program) from a set of positive and negative examples P and N. In particular, it is required that the hypothesis H covers all positive examples in P and none of the negative examples in N. The representation language for representing the examples together with the *covers* relation determines the ILP setting [56].

Learning from entailment is probably the most popular ILP setting and many well-known ILP systems such as FOIL [55], Progol [42] or ALEPH [66] follow this setting. In this setting examples are definite clauses and an example e is covered by an hypothesis H, w.r.t the background theory B if and only if $B \cup H \models e$. Most ILP systems in this setting require ground facts as examples. They typically proceed following a separate-and-conquer rule-learning approach [18]. This means that in the outer loop they repeatedly search for a rule covering many positive examples and none of the negatives (set-covering approach [40]). In the inner loop ILP systems generally perform a general-to-specific heuristic search using refinement operators [45, 63] based on θ-subsumption [49]. These operators perform the steps in the search-space, by making small modifications to a hypothesis. From a logical perspective, these refinement operators typically realize elementary generalization and specialization steps (usually under θ-subsumption). More sophisticated systems like Progol or ALEPH employ a search bias to reduce the search space of hypothesis.

In the ILP setting of learning from interpretations, examples are Herbrand interpretations and an examle e is covered by an hypthesis H, w.r.t the background theory B, if and only if e is a model of $B \cup H$. A possible world is described through sets of true ground facts which are the Herbrand interpretations. Learning from interpretations is generally easier and computationally more tractable than learning from entailment [56]. This is due to the fact that interpretations carry much more information than the examples in learning from entailment. In learning from entailment, examples consist of a single fact, while in interepretations all the facts that hold in the example are known. The approach followed by ILP systems learning from interpretations is similar to those that learn from entailment. The most important difference stands in the generality relationship. In learning from entailment an hypothesis H_1 is more general than H_2 if and only if $H_1 \models H_2$, while in learning from interpretations when $H_2 \models H_1$. A hypothesis H_1 is more general than a hypothesis H_2 if all examples covered by H_2 are also covered by H_1. ILP systems that learn from interpretations are also well suited for learning from positive examples only [10].

5 Abduction

In this section we present Abductive Logic Programing and how an abductive proof procedure can be integrated in an Inductice Logic Programming approach for incremental theory revision.

5.1 Abuctive Logic Programming

Abductive Logic Programming (ALP) [31, 15] is an extension of Logic Programming aimed at supporting abductive reasoning with theories (logic programs) that incompletely describe their problem domain. In ALP this incomplete knowledge is captured by an abductive theory, defined as a triple $(T,\mathscr{A},\mathscr{I})$ where T is a (hierarchical) logic program, $\mathscr{A}$ is a set of abducible predicates, and $\mathscr{I}$ is a set of integrity constraints represented as program clauses.

An abductive procedure can be exploited to deal with the problem of incompleteness by finding explanations that make hypotheses (abductive assumptions) on

Algorithm 1 Abductive Refutation Algorithm

```
**abduce**($T$, $G$, $\Delta$, $\mathscr{AD}$, $\mathscr{I}$)

{**input:** $T$: theory, $G$: Datalog goal (set of literals), $\Delta$: initial abductive assumptions, $\mathscr{AD}$: the
set of abducibles and default literals, $\mathscr{I}$: the integrity constraints;
**output:** $\Delta'$ final abductive assumptions;}
$\Delta' = \Delta$;
**while** $G \neq \emptyset$ **do**
  $L$ := Select a literal from $G$;
  **if** $L \notin \mathscr{AD}$ **then**
    /* (**A1**) */ $G$ := Resolvent of some clause of $T$ with G on L;
  **else if** $L \in \Delta'$ **then**
    /* (**A2**) */ $G := G \setminus L$;
  **else if** $\overline{L}_J \notin \Delta'$ and $\exists\Delta_C = consistency(T, L, \Delta' \cup \{L\}, \mathscr{AD}, \mathscr{I})$ **then**
    /* (**A3**) */ $G := G \setminus L$; $\Delta' := \Delta_C$;
  **end if**
**end while**
```

Algorithm 2 Consistency Derivation Algorithm

```
**consistency**($T$, $L$, $\Delta$, $\mathscr{AD}$, $\mathscr{I}$)

{**input:** $T$: theory, $L \in \mathscr{AD}$: a literal, $\Delta$: initial abductive assumptions,
$\mathscr{AD}$: the set of abducibles and default literals, $\mathscr{I}$: the integrity constraints;
**output:** $\Delta'$ final abductive assumptions;}
$\Delta' := \Delta$;
$C := \bigcup$ of goals of the form $: -L_1, L_2, \ldots, L_n$ obtained by resolving the abducibles or default
literal $L$ with the integrity constraints $\mathscr{I}$ with no such goal been empty;
**while** $C \neq \emptyset$ **do**
  $B$ := Select a goal from C; $M$ := Select a literal from B;
  **if** $M \notin \mathscr{AD}$ **then**
    $H$ := Resolvent of some clause of $T$ with B on M;
    $C := \{C \setminus B\} \cup H$;
  **else if** $M \in \mathscr{AD}$ and $M \in \Delta'$ **then**
    /* (**F1**) */ $H := B \setminus M$; $C := \{C \setminus B\} \cup H$;
  **else if** $M \in \mathscr{AD}$ and $\overline{M} \in \Delta'$ **then**
    /* (**F2**) */ $C := C \setminus B$;
  **else if** $M \in \mathscr{AD}$ and ($M \notin \Delta'$, $\overline{M} \notin \Delta'$) **then**
    /* (**F3**) */
    **if** $\exists\Delta_A = abduce(T, \overline{M}, \Delta', \mathscr{AD}, \mathscr{I})$ **then**
      $C := C \setminus B$; $\Delta' := \Delta_A$
    **end if**
  **end if**
**end while**
```

the state of the world, possibly involving new abducible concepts. The procedure is generally goal-driven by the observations that it tries to explain. Preliminarily, the top-level goal undergoes a transformation process that converts it into sub-goals. This provides a simple and unique modality for dealing with non-monotonic reasoning. Algorithm 1 sketches the classical abductive proof procedure proposed in [25]. After a literal is selected, if it is not abducible or a default one (A1), the procedure continues with a resolution step with clauses from T. Otherwise, if the fact has been already assumed abductively (and consistently) as true in previous steps (A2) it can be dropped (a case of successful proof). Otherwise (A3), a new fact may be assumed as true, provided that it is consistent with the current integrity constraints $\mathscr{I}$, which is verified by the consistency-check subroutine reported in Algorithm 2. The various branches in the consistency-check subroutine are similar to derivations except that, when dealing with an abducible or a default literal, if it has already been abduced (F1) then it is simply dropped (i.e. consistency is trivially proved); otherwise, if its complement has already been abduced or can be abduced (F2), the entire goal is dropped. In the last if-branch (F3), whenever the literal to be tested is an abducible or default one, but neither it nor its complement have been already abduced, the abductive procedure is called, in order to try hypothesizing it by abduction. Thus, the two procedures may call each other both when a new abductive assumption requires further consistency checks against the constraints and vice-versa.

Representing theories as *hierarchical* logic programs allows to maintain the Least Herbrand Models semantics, coping with negation by means of NAF [7] rule. Indeed, since the language of definite clauses with integrity constraints has been proven to subsume NAF [14], integrity constraints can be simulated using NAF as well. The advantage of adopting this semantics resides in the fact that $T \models P_1$, $T \models P_2$, $\dots$, $T \models P_n$ implies that $T \models P_1 \wedge P_2 \wedge \cdots \wedge P_n$. Hence, positive/negative examples can be tested separately for completeness/consistency.

5.2 Integrating Abductive Inference in Inductive Learning

Algorithm 3 sketches the integration of an incremental inductive learning framework with an abductive proof procedure as proposed in [16]. Here, M represents the set of all positive and negative processed examples, E is the example currently examined, T is the theory generated so far according to M, $AbdT$ is the abduction theory, D is the set of facts hypothesized by the abductive derivation when successfully applied to a goal in theory T. *Generalize* and *Specialize* are the inductive operators used by the system to refine an incorrect theory. When a new observation is available, the abductive proof procedure is started, parameterized on the current theory, the example and the current set of past abductive assumptions. If the procedure succeeds, the resulting set of assumptions, that were necessary to correctly classify the observation, is added to the example description, otherwise the usual refinement procedure (generalization/specialization) is performed.

Several aspects of the strategy adopted in Algorithm 3 can be useful for our purposes of learning the structure of an SRL model. First, it can be useful to apply the abductive derivation on examples that are not correctly classified (i.e., generate an omission/commission error) by the current theory using deduction only. The system checks whether the example can be correctly explained by hypothesizing new facts by means of the abductive procedure reported in Algorithm 1. Indeed, if successful, such an application provides abduced facts that can be useful for extending the available knowledge of the world. The incremental strategy exploits this feature to complete the observations in such a way that the corresponding examples are either covered (if positive) or ruled out (if negative) by the already generated theory, in order to avoid performing a revision of the theory whenever possible (only in case of failure the refinement operators are applied to modify/revise the theory). Abduction is thus exploited. The abductive proof procedure can be set in such a way that the set of abduced literals for each observation is minimal, which ensures that abducibles are used only when really needed, or maximal, which allows to make all possible consistent assumptions that can potentially provide new knowledge about the world. In [16] the minimal option is adopted in order to have a conservative behaviour, while here the maximal one could be more suitable to gain more information about the likelihood of the candidate theories. This is the approach that we follow here. Furthermore, in [16] the abduced information is attached directly to the observation that generated it, in order to keep observations independent from each other. However, this implies that the "completed" examples obtained this way must be available to subsequent abductions, so that the hypothesized facts can be considered in order to preserve consistency among the whole set of abduced facts. In our case, examples are to be exploited altogether, so there is no need to keep abductions attached to the corresponding observations, but a single initial goal including the conjunction of all available examples can be considered, which provides a unique set of abduced facts that explain the whole set of examples, are consistent among each other and can be exploited for the likelihood computation. Lastly, on the inductive side, another thing that can be borrowed is the exploitation of refinement operators that can modify a theory so that it can account for a new example on which it previously generated an omission/commission error. In our case, these operators can be exploited for guiding the move from a theory to one of its refinements, instead of randomly trying to apply all possible refinements.

6 Single Step Structure Learning with Abduction

In this section we describe how structure learning of MLNs in a single step can be combined with the procedure for logical abduction presented in the previous section. The algorithms we propose here are built upon the ideas that we presented in [4]. The parameters are set through maximum pseudo-log-likelihood (WPLL), and the structures are scored through conditional likelihood. The only difference regards the

Algorithm 3 Theory Revision extending an incremental inductive learning framework with an abductive proof procedure

```
**Revise** ($T$; $E$; $M$; $AbdT$);
{**input:** $T$: theory, $E$: example, $M$: historical memory, $AbdT$: Abductive Theory;
**output:** $T$ revised theory;}
$D \leftarrow E$
**if** ($Abductions$ = Abduct($T,E,D,AbdT$)) succeeds **then**
  Add to $D$ the abduced literals $Abductions$; $M \leftarrow M \cup \{E \cup D\}$;
**else** $M \leftarrow M \cup E$
  **if** $E$ is a positive example **then** Generalize($T,E,M$);
  **else** Specialize($T,E,M$);
```

use of logical abduction to complete unknown values in the data during structure search and before computing the WPLL score for each structure.

The first difference between the full framework of MLNs proposed by [58] and the framework that we propose here is that in order to use ALP during structure search we need to restrict the clauses of our model MLN to Horn clauses. Most of relational learning is performed under this expressiveness power and the successes of ILP have shown that for many problems Horn logic is sufficient to deal with structured domains. Thus the structure learning algorithms that we propose here are an extension of those proposed in [4, 5] in that here we perform logical abduction in the structure learning process and the language we follow here is based on Horn logic instead of full FOL. The second difference is that the algorithms proposed in [58], try to apply all possible refinements, while here we use ILP refinement operators to properly explore the search space.

6.1 Pseudo-likelihood

MLN weights can be learned by maximizing the likelihood of a relational database. Like in ILP, a closed-world assumption [19] is made, thus all ground atoms not in the database are assumed false. If there are n possible ground atoms, then we can represent a database as a vector $x = (x_1, ..., x_i ..., x_n)$ and x_i is the truth value of the ith ground atom, x_i = 1 if the atom appears in the database, otherwise x_i = 0. Standard methods can be used to learn MLN weights following Equation 4. If the jth formula has $n_j(x)$ true groundings, by Equation 4 we get the derivative of the log-likelihood with respect to its weights by:

$$\frac{\partial}{\partial w_j} logP_w(X = x) = n_j(x) - \sum_{x'} P_w(X = x')n_j(x') \tag{5}$$

where x' are databases and $P_w(X = x')$ is $P(X = x')$ computed using the current weight vector $w = (w_1, ..., w_j)$. Thus, the jth component of the gradient is the difference between the number of true groundings of the jth formula in the data and its expectation according to the model. Counting the number of true groundings of a first-order formula, unfortunately, is a #P-complete problem.

The problem with Equation 5 is that not only the first component is intractable, but also computing the expected number of true groundings is also intractable, requiring inference over the model. Further, efficient optimization methods also require computing the log-likelihood itself (Equation 4), and thus the partition function Z. This can be done approximately using a Monte Carlo maximum likelihood estimator (MC-MLE) [21]. However, the authors in [58] found in their experiments that the Gibbs sampling used to compute the MC-MLEs and gradients did not converge in reasonable time, and using the samples from the unconverged chains yielded poor results.

In many other fields such as spatial statistics, social network modeling and language processing, a more efficient alternative has been followed. This is optimizing pseudo-likelihood [3] instead of likelihood. If x is a possible world (a database or truth assignment) and x_l is the lth ground atom's truth value, the pseudo-likelihood of x is given by the following equation (we follow the same notation as the authors in [58]:

$$P_w^*(X=x)=\prod_{l=1}^{n} P_w(X_l=x_l|MB_x(X_l)) \tag{6}$$

where $MB_x(X_l)$ is the state of the Markov blanket of X_l in the data. (i.e., the truth values of the ground atoms it appears in some ground formula with). From Equation 4 we have:

$$P(X_l=x_l|MB_x(X_l))=\frac{\exp(\sum_{i=1}^{F} w_i n_i(x))}{\exp(\sum_{i=1}^{F} w_i n_i(x_{[X_l=0]}))+\exp(\sum_{i=1}^{F} w_i n_i(x_{[X_l=1]}))} \tag{7}$$

Or we can take the gradient of pseudo-log-likelihood:

$$\begin{aligned}\frac{\partial}{\partial w_i} logP_w^*(X=x)=\sum_{l=1}^{n}[n_i(x)-P_w(X_l=0|MB_x(X_l))n_i(x_{[X_l=0]})-\\ P_w(X_l=1|MB_x(X_l))n_i(x_{[X_l=1]})]\end{aligned} \tag{8}$$

where $n_i(x_{[X_l=1]})$ is the number of true groundings of the ith formula when $X_l=1$ and the remaining data do not change and similarly for $n_i(x_{[X_l=0]})$. To compute the expressions 7 or 8, we do not need to perform inference over the model. The optimal weights for pseudo-log-likelihood can be found using the limited-memory BFGS algorithm [34].

When computing $n_i(x_{[X_l=1]})$ and $n_i(x_{[X_l=0]})$, the usually followed approach is closed world assumption [19], i.e., all ground atoms not in the database are assumed false. Using logical abduction we can pontentially infer the truth value of these atoms and thus when we compute these counts we could have more accurate values that reflect the current data. Since the optimization of the weights by L-BFGS

Algorithm 4 MLNs Structure Learning with Abduction

```
Input: P:set of predicates, MLN:Markov Logic Network, RDB:Relational Database
CLS = All clauses in MLN;
LearnWPLLWeights(MLN,RDB);
BestScore = $f$(MLN,RDB);
BestModel = MLN;
repeat
  CurrentModel = FindBestModel(P,MLN,BestScore,CLS,RDB);
  if $f$(CurrentModel) $\geq f$(BestModel) then
    BestModel = CurrentModel;
    BestScore = $f$(MLN,RDB);
  end if
until BestScore does not improve for two consecutive steps
return BestModel;
$f$ = CLL (conditional log-likelihood)
```

is performed on the estimates of the counts $n_i(x_{[X_l=1]})$ and $n_i(x_{[X_l=0]})$, an improved accuracy on these counts would also result in a more accurate parameter learning task. Thus the use of logical abduction is motivated by the fact that parameter estimation in satistical relational learning can benefit from completed data through logical procedures. To the best of our knowledge, this is the first approach to integrate a pure logical procedure for abductive inference with a statistical parameter estimation algorithm.

6.2 Structure Learning with Abduction

Structure learning can start from an empty network or from an existing KB. Algorithm iteratively generates refinements of the current structure and scores them by conditional likelihood. These refinements are generated using normal ILP refinement operators. Every neighbor of the current structure is obtained by a small generalization/specialization of a randomly chosen clause in the structure. Algorithm 5 performs Iterated Local Search [23, 35] for the best model that fits the data. It starts by randomly choosing a clause CL_C in the current MLN structure. Then it performs a greedy local search to efficiently reach a local optimum MLN_S. At this point, a restart method is applied by randomly choosing a clause CL'$_C$ from the clauses of MLN_S. Then again, a greedy local search is applied to MLN_S to reach another local optimum MLN'_S . The *accept* function decides whether the search must continue from the previous local optimum MLN_S or from the last local optimum MLN'_S. The *accept* function always accepts the best solution found so far.

For every candidate structure, the parameters that optimize the WPLL are set through L-BFGS. As pointed out in [29] a potentially serious problem that arises when evaluating candidate clauses using WPLL is that the optimal (maximum WPLL) weights need to be computed for each candidate. Since this involves numerical optimization, and needs to be done millions of times, it could easily make

Algorithm 5 FindBestModel

```
Input: P:set of predicates, MLN:Markov Logic Network, BestScore: current best score, CLS:
List of clauses, RDB:Relational Database)
$CL_C$ = Random Pick a clause in CLS;
$MLN_S$ = $LocalSearch_{II}$($CL_C$,MLN,BestScore);
BestModel = $MLN_S$;
repeat
  $CL'_C$ = Random Pick a clause in ($MLN_S$);
  $MLN'_S$ = $LocalSearch_{II}$($CL'_C$,$MLN_S$,BestScore);
  if $f$(BestModel,RDB) $\geq f$($MLN'_S$,RDB) then
    BestModel = $MLN'_S$;
    BestScore = $f$($MLN'_S$,RDB)
  end if
  $MLN_S$ = accept($MLN_S$,$MLN'_S$);
until two consecutive steps have not produced improvement
Return BestModel
$f$ = CLL (conditional log-likelihood)
```

the algorithm too slow. In [37, 12] the problem is addressed by assuming that the weights of previous features do not change when testing a new one. Surprisingly, the authors in [29] found this to be unnecessary if the very simple approach of initializing L-BFGS with the current weights (and zero weight for a new clause) is used. Although in principle all weights could change as the result of introducing or modifying a clause, in practice this is very rare. Second-order, quadratic-convergence methods like L-BFGS are known to be very fast if started near the optimum [62]. This is what happened in [29]: L-BFGS typically converges in just a few iterations, sometimes one. We use the same approach for setting the parameters that optimize the WPLL.

In Algorithm 6, we generate NBHD, the neighborhood of MLN_C, by using ILP refinement operators. All structures in NBHD differ from MLN_C by only one clause which is a generalization or specialization of the clause CL_C. Two modifications can be applied here with respect to the traditional setting. First of all, the structure refinement is not carried out randomly, but can be guided by the examples themselves, since they were purposely provided by an expert. Hence, each example that is not correctly classified by the current theory can be exploited to perform a generalization (if positive) or specialization (if negative) according to classical ILP operators. Application of such an operator will provide one or more (depending on the operator and on the generalization model adopted) alternative refinements of the original structure, each of which consists in a new structure obtained by refining a single clause in the original structure. Moreover, pruning criteria can be set in order to avoid working on refinements that are not regarded as promising or acceptable. For instance, one could require that each candidate structure fulfils a minimum coverage threshold in the logical sense, i.e., that the accuracy from the ILP point of view (how many positive examples are covered and how many negatives are not) is greater than a given minimum. We believe this heuristic can help exclude candidates that have a very low logical accuracy. Although there is a mismatch between the coverage

Algorithm 6 LocalSearch$_{II}$

```
Input: (CL_C: clause chosen for refinement, MLN_C: current model, BestScore: current best score)
wp: walk probability, the probability of performing an improvement step or a
random step
repeat
  NBHD = Neighborhood of MLN_C constructed using ILP refinement operators on the clause CL_C;
  for Each Candidate Structure MLN in NBHD do
    if MLN satisfies ILP coverage threshold then
      PerformLogicalAbduction(MLN,RDB);
      if all atoms have known truth values then
        LearnWPLLWeights(MLN,RDB);
      else
        LearnWPLLWeightswithEM(MLN,RDB);
      end if
    end if
  end for
  for Each structure scored MLN do
    score = $f$(MLN,RDB)
    if score $\geq$ BestScore then
      BestScore = score;
      MLN_S = MLN
    end if
  end for
until two consecutive steps do not produce improvement
Return MLN_S;
```

criterion used by most ILP systems and the likelihood (or a function thereof) used by most statistical learners, a logical theory that does not explain any example from a logical interpretation would be less useful, and would contradict the idea that examples are purposely labelled by an expert and hence deserve some level of trust. Therefore, we decided to pose a threshold on the accuracy of candidate structures and learn weights only for those candidates that satisfy this threshold.

After the coverage check, we perform logical abduction using the theory of each structure and the examples in RDB. When the abductive proof procedure has potentially completed missing values in RBD, we check whether all the data have been completed. If this is the case then we can learn optimal WPLL weights without EM, otherwise we use EM. For very incomplete data, it is probable that the abductive proof procedure will not complete all the missing data. However, the partial completing of the data will potentially help the weight learning procedure to learn more accurate weights compared to the case when more data is missing.

After setting weights with WPLL, in order to score each MLN structure in terms of conditional likelihood (CLL), we need to perform inference over the network. A very fast algorithm for inference in MLNs is MC-SAT [52]. Since probabilistic inference methods like MCMC or belief propagation tend to give poor results when deterministic or near-deterministic dependencies are present, and logical ones like satisfiability testing are inapplicable to probabilistic dependencies, MC-SAT combines ideas from both MCMC and satisfiability to handle probabilistic, deterministic and near-deterministic dependencies that are typical of statistical relational learning. MC-SAT was shown to greatly outperform Gibbs sampling and simulated tempering in two real-world datasets regarding entity resolution and collective classification. MC-SAT produces probability outputs for every grounding of the query predicate on the test fold and these values can be used to compute the average CLL over all the groundings. In order to make the execution of MC-SAT tractable for every candidate structure, we follow the same heuristic that were proposed in [4], i.e., we score through MC-SAT only those candidate structures that show an improvement in WPLL, we use the lazy version of MC-SAT that is known as Lazy-MC-SAT [53] which reduces memory and time by orders of magnitude compared to MC-SAT, we pose a memory and time limit on the inference process thorugh Lazy-MC-SAT. As the experiments showed in [4], these heuristics proved quite successful in two real world domains. We denote this framework as Structure Learning with Abduction (SLA).

7 Structural EM with Abduction

In the presence of missing values a procedure normally used is Expectation- Maximization (EM) [13]. In this section we describe the EM algorithm and the Structural-EM algorithm that was first proposed in [17] to learn the structure of Bayesian Networks. Then we sketch a framework for integrating logical abduction in the Structural-EM algorithm and discuss the benefits that the statistical learning setting can have from logical abduction.

7.1 Expectation-Maximization and Structural EM

In the presence of missing values maximum likelihood parameter estimation is a numerical optimization problem, and all known algorithms involve nonlinear, iterative optimization and multiple calls to an inference algorithm. The most widely used algorithm for parameter estimation under hidden variables is Expectation-Maximization [13]. This algorithm proceeds in two steps, in the Expectation (E)-step it is computed the expectation of the previous model and the observed data and in the Maximization (M) step, the expected score is maximized. Thus, if we denote the previous model MLN_k and the parameters of the model $\lambda_{k,l}$ in the l step, then in the $l+1$ the algorithm performs two steps:

E-Step: Computes the expectation of the log-likelihood given the old model $(MLN_k, \lambda_{k,l})$ and the observed data D, i.e., $Q(MLN_k, \lambda | MLN_k, \lambda_{k,l})$
$= E[logP(D|MLN_k, \lambda)|MLN_k, \lambda_{k,l}]$.

Algorithm 7 Structural EM

```
Input: (Current model: $MLN_k$, $\lambda_{k,l}$, RDB:relational data)
Perform random assignment of $\lambda_{0,0}$
repeat
  for k = 0, 1, 2, . . .
  repeat
    for l = 0, 1, 2, . . .
    $\lambda_{k,l+1} = argmaxQ(MLN_k, \lambda | MLN_k, \lambda_{k,l})$
  until convergence is reached or $l = l_{max}$
  Find a model $MLN_{k+1} \in$ neighbors($MLN_k$) that maximizes
  $max_\lambda Q(MLN_{k+1}, \lambda | MLN_k, \lambda_{k,l})$
  Set $\lambda_{k+1,0} = argmaxQ(MLN_{k+1}, \lambda | MLN_k, \lambda_{k,l})$
until convergence is reached
neighbors($MLN_k$) is computed using the ILP refinement operators.
```

where D denotes the completion of the data. The current model MLN_k, $\lambda_{k,l}$ and the observed data D give the conditional distribution and E denotes the expectation over it. The function Q is called the *expected score*.

M-Step: Maximize the expected score $Q(MLN_k, \lambda | MLN_k, \lambda_{k,l})$ w.r.t. λ, i.e., $\lambda_{k,l+1} = argmax_\lambda Q(MLN_k, \lambda | MLN_k, \lambda_{k,l})$.

Algorithm 4 can be instantiated using the EM. The problem, however, is the huge computational costs. To evaluate a single neighbor, the EM has to run for a reasonable number of iterations in order to get reliable ML estimates of λ_k . Each EM iteration requires a full inference on all data cases. In total, the running time per a neighbor evaluation is at least O(#EM iterations * size of data) which is intractable even for very simple problems. The idea of Structural EM [17] is to perform structure search inside the EM procedure. Algorithm 7 takes the current model $(MLN_k, \lambda_{k,l})$ and runs the EM algorithm for a while to get reasonably completed data. It then fixes the completed data cases and used them to compute the ML parameters λ_k of each neighbor MLN_k. The neighbor $(MLN_{k+1}, \lambda_{k+1})$ with the best improvement of the score is chosen for the next iteration.

7.2 *Integrating Logical Abduction in Structural EM*

Algorithm 8 shows how the abductive proof procedure can be plugged in the Structual EM algorithm. The logical abduction process is performed inside the E-step, in order to complete the available data. After the abductive process is completed, the EM approach fixes the current model $(MLN_k, \lambda_{k,l})$ and computes maximum pseudo-likelihood parameters of the neighbors of $(MLN_k, \lambda_{k,l})$. The neighbors are constructed using the ILP refinement. After weights have been set for each neighbor, the average CLL for each structure is then computed based on these weights using

Algorithm 8 Structural EM with abduction

```
Input: (Current model: $MLN_k$, $\lambda_{k,l}$, RDB:relational data)
Perform random assignment of $\lambda_{0,0}$
repeat
  for k = 0, 1, 2, . . .
  CL$_C$ = Random Pick a clause in $MLN_k$;
  NBHD = Neighborhood of MLN$_k$ constructed using ILP refinement operators on the clause CL$_C$;
  repeat
    for l = 0, 1, 2, . . .
    PerformLogicalAbduction($MLN_k$,RDB);
    $\lambda_{k,l+1} = argmaxWPLLQ(MLN_k, \lambda | MLN_k, \lambda_{k,l})$
  until convergence is reached or $l = l_{max}$
  Find a model $MLN_{k+1} \in$ NBHD that maximizes
  $maxWPLL_{\lambda} Q(MLN_{k+1}, \lambda | MLN_k, \lambda_{k,l})$
  score each structure with CLL using MC-SAT
  Set $\lambda_{k+1,0} = argmaxCLLQ(MLN_{k+1}, \lambda | MLN_k, \lambda_{k,l})$
until convergence is reached
```

MC-SAT. The best model is the one that maximizes CLL. We call this framework Structural EM with Logical Abduction (SEMLA).

The difference with the framework proposed in the previous section is that the abductive proof procedure in SEMLA is executed on the current model trying to complete the data based on the current theory. While in SLA the logical abductive process is performed on each of the neighbors of the current model exploiting a different theory which is obtained by refinement operators from the current theory. Another difference of SLA and SEMLA is that in SEMLA the E-step is performed for the current model with the current parameters, while in SLA the E-step is performed on each candidate structure separately with a different set of parameters. Finally, the M-step for SEMLA is performed on all the neighbors of the current model using the estimates on the current model and trying to maximize the likelihood of each neighbor, while in SLA the M-step uses the independent estimates on each of the candidate structures to maximize its likelihood.

From a computational complexity point of view, we expect SLA to be more expensive since logical abduction and the entire E-step is performed for each of the neighbors of the current model, while for SEMLA, both logical abduction and the E-step are performed only once for the current model and then used for all the neighbors. However, since the abduced atoms change with the available theory, in SLA the logical abduction process would produce abducibles according to the logical theory of each neighbor, thus the abduced truth values are directly related to each neighbor. In SEMLA this is not the case, for each neighbor the abduced truth values with the current model are used and these values may not be directly related with the theory of the neighbor.

8 Related Work

To the best of our knowledge this is the first proposal that tightly integrates in a task of structure learning algorithm, a logical approach to abduction with a statistical procedure for parameter learning. Previous related work has considered mostly statistical abduction as the principal form of inference and has considered logic simply as a representation formalism. One of the first approaches similar to ours is that of [51] where *probabilistic Horn abduction* was proposed. In this approach, a program contains non-probabilistic definite clauses and probabilistic disjoint declarations which are of the form $h_1 : p_1, ..., h_n : p_n$ and an abducible atom h_i is considered true with probability p_i. This work focuses on the representation language issue trying to propose a simple language for integrating logic and probability and the authors does not deal with the structure learning problem. Moreover, it does not integrate any form of logic-based abductive proof procedure with statistical learning. Another approach is that in [61], where a logic-based framework is proposed and *statistical abduction* is introduced for representing and learning probabilistic knowledge. The abductive inference is made possible through the definiton of a probability distribution over abducibles. This makes possible to identify the best hypothesis as the most likely hypothesis and likelihood is maximized through statistical learning. The difference with our proposal is that the approach of [61] is purely statistical and the role of logic is purely sintactic, i.e., there is no pure logic-based proof procedure as in our two proposed frameworks. Moreover, the authors in [61] do not learn the structure of the model as we do here. Instead, they hand code the clauses of the model and only learn the statistical parameters of the model through an EM based algorithm. Finally, our SEMLA framework modifies the EM algorithm in a way that structure search can be performed inside the EM algorithm together with logical abduction. Finally, a similar approach is that proposed in [1, 6] where the authors proposed Abductive Stochastic Logic Programs which is a framework that supports abduction in SLPs [41] to provide a probability distribution over abductive hypothesis based on a *possible world* semantics. Again the main difference with our proposed frameworks is that the approaches in [1, 6] suppose to have an already learned structure in order to learn the parameters for the SLP. When the parameters of the SLP have been learned, this probabilistic program is used to define a probability distribution over the abducibles using a stochastic SLD derivation. The labelled hypothesised abducibles are chosen to maximize the likelihood. Therefore, since the structure of the model is first learned by ILP using “coverage” as guiding function, and all the following process involves only parameter learning, our proposals are different since we learn the structure by directly optimizing a likelihood based function. For an SRL this has proven to be the best way to learn a model as the results of [29] show, where ILP based approaches were outperformed by likelihood-guided approaches for the task of learning the structure of an SRL model. Moreover, we perform abduction during structure selection, while the approach of [1, 6] uses a two step approach, first learns the structure with ILP (then the parameters) and then performs abduction. This two step approach has been shown in [32] to be inferiror in terms of accuracy compared to the single-step structure learning approach that we follow here.

9 Conclusion and Future Work

Statistical Relational Learning (SRL) is a growing field in Machine Learning that aims at the integration of logic-based learning approaches with probabilistic graphical models. Markov Logic Networks are one of the state-of-the-art SRL models that combine first-order logic and Markov networks (MNs) by attaching weights to first-order formulas and viewing these as templates for features of MNs. Learning models in SRL consists in learning the structure (logical clauses in MLNs) and the parameters (weights for each clause in MLNs). Structure learning of MLNs is performed by maximizing a likelihood function (or a function thereof) over relational databases and MLNs have been successfully applied to problems in relational and uncertain domains. However, most complex domains are characterized by incomplete data. Until now SRL models have mostly used Expectation-Maximization for learning statistical parameters under missing values. Multistrategic learning in the relational setting has been a successful approach to dealing with complex problems where multiple inference mechanisms can help solve different subproblems. Abduction is an inference strategy that has been proven useful for completing missing values in observations. In this paper we propose two frameworks for integrating abduction in an SRL model based on MLNs. The first tightly integrates logical abduction with structure and parameter learning of MLNs in a single step. During structure search guided by conditional likelihood, clause evaluation is performed by first trying to logically abduce missing values in the data and then by learning optimal parameters using the completed data. The second approach integrates abduction with Structural EM of [17] by performing logical abductive inference in the E-step and then by trying to maximize parameters in the M-step.

We intend to experimentally evaluate the proposed frameworks on complex relational domains with missing data. In order to evaluate the advantages of our approach, we intend to compare the accuracy performance against a pure statistical learner that uses EM to deal with missing values, a pure logical approach such as an ILP system and finally against another SRL approach that does not follow our approach to structure learning.

References

1. A. Arvanitis, S.H. Muggleton, J. Chen, and H. Watanabe. Abduction with stochastic logic programs based on a possible worlds semantics. In *In Short Paper Proceedings of the 16th International Conference on Inductive Logic Programming.* University of Corunna, 2006.
2. F. Bacchus. *Representing and Reasoning with Probabilistic Knowledge*. Cambridge, MA: MIT Press, 1990.

3. J. Besag. Statistical analysis of non-lattice data. *Statistician*, 24:179–195, 1975.
4. M. Biba, S. Ferilli, and F. Esposito. Discriminative structure learning of markov logic networks. In *Proceedings of 18th International Conference on Inductive Logic Programming, (ILP 2008), LNCS 5194*, pages 59–76. Springer, 2008.
5. M. Biba, S. Ferilli, and F. Esposito. Structure learning of markov logic networks through iterated local search. In *Frontiers in Artificial Intelligence and Applications, Proceedings of 18th European Conference on Artificial Intelligence (ECAI).*, volume 178, pages 361–365, 2008.
6. J. Chen, S. Muggleton, and J. Santos. Abductive stochastic logic programs for metabolic network inhibition learning. In *In Proceedings of Workshop Mining and Learning with Graphs, MLG07*, 2007.
7. K. Clark. Negation as failure. In *Gallaire H, Minker J (eds) Logic and databases*, pages 293–322. Plenum Press, New York, 1978.
8. C. Cumby and D. Roth. Feature extraction languages for propositionalized relational learning. In *Proceedings of the IJCAI-2003 Workshop on Learning Statistical Models from Relational Data*, pages 24–31. Acapulco, Mexico: IJCAII, 2003.
9. J. Cussens. Parameter estimation in stochastic logic programs. *Machine Learning*, 44(3):245–271, 2001.
10. L. De Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 26:99–146, 1997.
11. L. De Raedt, P. Frasconi, K. Kersting, and S. Muggleton, editors. *Probabilistic Inductive Logic Programming - Theory and Applications*. Springer, 2008.
12. S. Della Pietra, V. Della Pietra, and J. Laferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–392, 1997.
13. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, Series B, vol. 39:1–38, 1977.
14. K. Eshghi and R. Kowalski. Abduction compared to negation by failure. In *Levi, G. and Martelli, M. (eds) Proceedings of the 6th international conference on logic programming*, page 234255. The MIT Press, Cambridge, MA, 1989.
15. F. Esposito, E. Lamma, P. Malerba, D.and Mello, M. Milano, F. Riguzzi, and G. Semeraro. Learning abductive logic programs. In *Proceedings of the ECAI96 workshop on abductive and inductive reasoning, Budapest*, pages 23–30, 1996.
16. F. Esposito, G. Semeraro, N. Fanizzi, and S. Ferilli. Multistrategy theory revision: induction and abduction in inthelex. *Machine Learning*, 38(1-2):133156, 2000.
17. N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In *Fourteenth Inter. Conf. on Machine Learning (ICML97)*, 1997.
18. J. Furnkranz. Separate-and-conquer rule learning. *Articial Intelligence Review*, 13(1):3–54, 1999.
19. M. R. Genesereth and N. J. Nilsson. *Logical foundations of artificial intelligence*. San Mateo, CA: Morgan Kaufmann., 1987.
20. L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning*. MIT, 2007.
21. C. J. Geyer and E. A. Thompson. Constrained monte carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society*, Series B, 54:657–699, 1992.
22. J. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1990.
23. H. H. Hoos and T. Stutzle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, San Francisco, 2005.
24. T. N. Huynh and R. J. Mooney. Discriminative structure and parameter learning for markov logic networks. In *In Proc. of the 25th International Conference on Machine Learning (ICML)*, 2008.
25. A. Kakas and P. Mancarella. On the relation of truth maintenance and abduction. In *Proc. 1st Pacic Rim International Conference on Articial Intelligence*, 1990.
26. A. Kakas and F. Riguzzi. Learning with abduction. *New Generation Computing*, 18(3):243–294, 2000.
27. M. Kakas, R. Kowalski, and F. Toni. Abductive logic programming. *J. Logic. Comput.*, pages 718–770, 1993.

28. K. Kersting and L. De Raedt. Towards combining inductive logic programming with bayesian networks. In *Proc. 11th Int'l Conf. on Inductive Logic Programming*, pages 118–131. Springer, 2001.
29. S. Kok and P. Domingos. Learning the structure of markov logic networks. In *Proc. 22nd Int'l Conf. on Machine Learning*, pages 441–448, 2005.
30. D. Koller, A. Levy, and A. Pfeffer. P-classic: A tractable probabilistic description logic. In *In Proc. of NCAI97*, pages 360–397, 1997.
31. E. Lamma, P. Mello, M. Milano, F. Riguzzi, F. Esposito, S. Ferilli, and G. Semeraro. *Abductive and inductive reasoning: essays on their relation and integration*, chapter Cooperation of abduction and induction in logic programming. Kluwer, Dordrecht, 2000.
32. N. Landwehr, K. Kersting, and L. De Raedt. Integrating naive bayes and foil. *Journal of Machine Learning Research*, pages 481–507, 2007.
33. N. Lavrac and S. Dzeroski. *Inductive Logic Programming: Techniques and applications*. UK: Ellis Horwood, Chichester, 1994.
34. D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.
35. H.R. Loureno, O. Martin, and T. Stutzle. Iterated local search. In *Handbook of Metaheuristics*, pages 321–353. F. Glover and G. Kochenberger, Kluwer Academic Publishers, Norwell, MA, USA, 2002.
36. D. Lowd and P. Domingos. Efficient weight learning for markov logic networks. In *Proc. of the 11th PKDD*, pages 200–211. Springer Verlag, 2007.
37. A. McCallum. Efficiently inducing features of conditional random fields. In *Proc. UAI-03*, pages 403–410, 2003.
38. R.S. Michalski. Inferential theory of learning. developing foundations for multistrategy learning. In *In R.S. Michalski and G. Tecuci, editors, Machine Learning. A Multistrategy Approach, volume IV, pages*, page 361. Morgan Kaufmann.
39. L. Mihalkova and R. J. Mooney. Bottom-up learning of markov logic network structure. In *Proc. 24th Int'l Conf. on Machine Learning*, pages 625–632, 2007.
40. T. M. Mitchell. *Machine Learning*. The McGraw-Hill Companies, Inc., 1997.
41. S. Muggleton. Stochastic logic programs. In *In L. De Raedt (Ed.), Advances in inductive logic programming*. IOS Press, Amsterdam, 1996.
42. S. H. Muggleton. Inverse entailment and progol. *New Generation Computing Journal*, pages 245–286, 1995.
43. D. Koller N. Friedman, L. Getoor and A. Pfeffer. Learning probabilistic relational models. In *Proc. 16th Int'l Joint Conf. on AI (IJCAI)*, pages 1300–1307. Morgan Kaufmann, 1999.
44. L. Ngo and P. Haddawy. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171:147–177, 1997.
45. S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*. Springer-Verlag, 1997.
46. N. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.
47. H. Pasula and S. Russell. Approximate inference for rst-order probabilistic languages. In *Proceedings of the Seventeenth International Joint Conference on Articial Intelligence*, pages 741–748. Seattle, WA: Morgan Kaufmann, 2001.
48. J. Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Francisco, CA: Morgan Kaufmann, 1988.
49. G. D. Plotkin. A note on inductive generalization. *In Machine Intelligence, Edinburgh University Press*, 5:153–163, 1970.
50. D. Poole. A logical framework for default reasoning. *Artif Intell*, 36:27–47, 1988.
51. D. Poole. Probabilistic horn abduction and bayesian networks. *Articial Intelligence*, 64(81-129), 1993.
52. H. Poon and P. Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proc. 21st Nat'l Conf. on AI, (AAAI)*, pages 458–463. AAAI Press, 2006.
53. H. Poon, P. Domingos, and M. Sumner. A general method for reducing the complexity of relational inference and its application to mcmc. In *Proc. 23rd Nat'l Conf. on Articial Intelligence*. Chicago, IL: AAAI Press, 2008.

54. A. Popescul and L. H. Ungar. Structural logistic regression for link analysis. In *Proceedings of the Second International Workshop on Multi-Relational Data Mining*, pages 92–106. Washington, DC: ACM Press, 2003.
55. J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.
56. L. De Raedt. Logical settings for concept-learning. *Articial Intelligence*, 95(1):197–201, 1997.
57. R. Reiter. A logic for default reasoning. *J. Artif. Intell.*, (13):81–132, 1980.
58. M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–236, 2006.
59. V. Santos Costa, D. Page, M. Qazi, and J. Cussens. Clp(bn): Constraint logic programming for probabilistic knowledge. In *Proceedings of the Nineteenth Conference on Uncertainty in Articial Intelligence*, pages 517–524. Acapulco, Mexico: Morgan Kaufmann, 2003.
60. T. Sato and Y. Kameya. Prism: A symbolic-statistical modeling language. In *Proceedings of the Fifteenth International Joint Conference on Articial Intelligence*, pages 1330–1335. Nagoya, Japan: Morgan Kaufmann, 1997.
61. T. Sato and Kameya Y. A viterbi-like algorithm and em learning for statistical abduction. In *Proceedings of UAI2000 Workshop on Fusion of Domain Knowledge with Data for Decision Support*, 2000.
62. F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proc. HLT-NAACL-03*, pages 134–141, 2003.
63. E. Shapiro. *Algorithmic Program Debugging*. MIT Press, 1983.
64. P. Singla and P. Domingos. Discriminative training of markov logic networks. In *Proc. 20th Nat'l Conf. on AI, (AAAI)*, pages 868–873. AAAI Press, 2005.
65. P. Singla and P. Domingos. Markov logic in infinite domains. In *Proc. 23rd UAI*, pages 368–375. AUAI Press, 2007.
66. A. Srinivasan. *The Aleph Manual.* Available at http://www.comlab.ox.ac.uk/oucl/ esearch/areas/machlearn/Aleph/.
67. B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth Conference on Uncertainty in Articial Intelligence*, pages 485–492. Edmonton, Canada: Morgan Kaufmann, 2002.
68. J. S. Wellman, M. Breese and R. P. Goldman. From knowledge bases to decision models. *Knowledge Engineering Review*, 7, 1992.