Extending Bayesian Logic Programs for Plan Recognition and Machine Reading

Sindhu V. Raghavan
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
sindhu@cs.utexas.edu

Doctoral Dissertation Proposal

Supervising Professor: Raymond J. Mooney

Abstract

Statistical relational learning (SRL) is the area of machine learning that integrates both first-order logic and probabilistic graphical models. The advantage of these formalisms is that they can handle both uncertainty and structured/relational data. As a result, they are widely used in domains like social network analysis, biological data analysis, and natural language processing. Bayesian Logic Programs (BLPs), which integrate both first-order logic and Bayesian networks are a powerful SRL formalism developed in the recent past. In this proposal, we focus on applying BLPs to two real worlds tasks – plan recognition and machine reading.

Plan recognition is the task of predicting an agent's top-level plans based on its observed actions. It is an abductive reasoning task that involves inferring cause from effect. In the first part of the proposal, we develop an approach to abductive plan recognition using BLPs. Since BLPs employ logical deduction to construct the networks, they cannot be used effectively for plan recognition as is. Therefore, we extend BLPs to use logical abduction to construct Bayesian networks and call the resulting model Bayesian Abductive Logic Programs (BALPs). Experimental evaluation on three benchmark data sets demonstrate that BALPs outperform the existing state-of-art methods like Markov Logic Networks (MLNs) for plan recognition.

For future work, we propose to apply BLPs to the task of machine reading, which involves automatic extraction of knowledge from natural language text. Present day information extraction (IE) systems that are trained for machine reading are limited by their ability to extract only factual information that is stated explicitly in the text. We propose to improve the performance of an off-the-shelf IE system by inducing general knowledge rules about the domain using the facts already extracted by the IE system. We then use these rules to infer additional facts using BLPs, thereby improving the recall of the underlying IE system. Here again, the standard inference used in BLPs cannot be used to construct the networks. So, we extend BLPs to perform forward inference on all facts extracted by the IE system and then construct the ground Bayesian networks. We initially use an existing inductive logic programming (ILP) based rule learner to learn the rules. In the longer term, we would like to develop a rule/structure learner that is capable of learning an even better set of first-order rules for BLPs.

Contents

1 Introduction

Statistical relational learning (SRL) (Getoor & Taskar, 2007) is the area of machine learning that integrates first-order logic and probabilistic graphical models. These SRL formalisms are useful for solving problems that involve relational or structured data. Let us consider the example of predicting the task that a user is performing on a computer based on the actions performed by him/her. The user could be performing the task of copying a file or moving a file from one directory to another, and he could be working on several different files at the same time. The files and directories represent different entities and the tasks the user is performing represent different relations between those entities. Now, the prediction task involves inferring not only the correct relation, but also the entities that participate in the relation. Traditional statistical learning techniques like Bayesian networks or Markov networks cannot be used for such problems as these models are essentially propositional in nature. Even though pure first-order logic based approaches handle structured data, they cannot handle uncertainty. But there is always uncertainty in real data - uncertainty in the relations between different entities, uncertainty in the types of entities, etc. In order to overcome these limitations, SRL formalisms combine strengths of both first-order logic and statistical models, making them useful for solving problems with relational/structured data.

Because of their advantages, SRL formalisms are widely used for social network analysis (e.g. (Richardson & Domingos, 2006)), biological data analysis (e.g. (Perlich & Merugu, 2005; Huynh & Mooney, 2008)), information extraction (e.g. (Bunescu & Mooney, 2007)), and other domains that involve structured/relational data. As a result, the last few years have seen a development of several SRL formalisms like Probabilistic Relational Models (PRMs) (Friedman, Getoor, Koller, & Pfeffer, 1999), Stochastic Logic Programs (SLPs) (Muggleton, 2000), Bayesian Logic Programs (BLPs) (Kersting & De Raedt, 2001, 2007), Markov Logic Networks (MLNs) (Richardson & Domingos, 2006) etc. BLPs, which integrate first-order logic and Bayesian networks are a simple, yet powerful formalism for solving problems with structured data. One advantage of BLPs over other SRL formalisms like MLNs is with regard to the grounding process used to construct Bayesian networks. Unlike MLNs, BLPs do not include all possible groundings of a rule in the ground network. Instead, they include only those rules that are relevant to the query. As a result, BLPs scale well to large domains. Further, due to the directed nature of BLPs, it is possible to use any type of logical inference to construct the networks. Finally, since Bayesian networks are a mature technology, a lot of existing machinery developed for Bayesian networks like algorithms for probabilistic inference can be used for BLPs as well. Because of these reasons, we have chosen to use BLPs in our research. In this proposal, we focus on applying BLPs to two real world tasks – plan recognition and machine reading.

Plan recognition is the task of predicting an agent's top-level plans based on its observed actions. Since it is an abductive reasoning task that involves inferring cause from effect (Charniak & McDermott, 1985), we believe that a directed probabilistic logic like BLPs will be better suited for plan recognition than an undirected probabilistic logic like MLNs. In the first part of the proposal, we develop an approach to abductive plan recognition using BLPs. BLPs use SLD resolution to generate proof trees, which are then used to construct a ground Bayes net for a given query. However, deduction is unable to construct proofs for abductive problems such as plan recognition because deductive inference involves predicting effects from causes, while the plan recognition task involves inferring causes (top-level plans) from effects (observations). Therefore, we extend BLPs to use logical abduction to construct proofs. In logical abduction, missing facts are assumed when necessary to complete proof trees, and we use the resulting abductive proof trees to construct Bayes nets. We call the resulting model Bayesian Abductive Logic Programs (BALPs) (Raghavan & Mooney, 2010). We learn the parameters for the BALP framework automatically from data using the Expectation Maximization algorithm adapted for BLPs by Kersting and De Raedt (2008). Experimental evaluation on three benchmark data sets demonstrate that BALPs outperform the existing state-of-art methods like MLNs for plan recognition.

In the second part of the proposal, we propose to apply BLPs to the task of machine reading, which involves automatic extraction of knowledge from unstructured text. Present day information extraction (IE) systems are trained to extract only factual information that is stated explicitly in the text. Typically, in natural language text, several facts or commonsense information are omitted from the text and some queries require this type of implicit information to be returned as answers. Due to this reason, we find that traditional IE systems are limited in their ability to answer queries. We propose to improve the performance of an off-the-shelf IE system by inducing general knowledge rules about the domain using the facts already extracted by the IE system. We then use these rules to infer additional facts using BLPs, thereby improving the recall of the underlying IE system. Here again, the standard inference used in BLPs cannot be used to construct the required networks. So, we extend BLPs to perform forward inference on all facts extracted by the IE system and then construct the ground Bayesian networks. In our preliminary analysis, we demonstrate that the first-order rules or the structure of BLP can be induced using LIME (Mccreath & Sharma, 1998), an existing inductive logic programming (ILP) based rule learner. We specify the parameters of the BLP model manually. We then perform BLP inference based on the facts extracted by the IE system to infer additional facts.

In the short-term, we propose to extend our work in the following directions:

- *Parameter learning and evaluation of BLPs for machine reading*
  We propose to automatically learn the parameters of the BLP model from data and perform an extensive evaluation of the BLP approach by comparing it with the state-of-the-art for machine reading like MLNs. To learn the parameters of BLP, we propose to use the EM algorithm adapted for BLPs by Kersting and De Raedt (2008). Lack of ground truth poses a challenge for evaluation in the machine reading task. For a given piece of natural language text, we do not have access to all the facts present in the text. As a result, even if our system infers an additional fact, it is not possible to evaluate it automatically. In order to evaluate the performance of our system, we propose to use human evaluation using Amazon Mechanical Turk (Callison-Burch & Dredze, 2010).

- *Structure learning with incomplete and uncertain data*
  Existing methods (Kersting & Raedt, 2008; Mihalkova & Mooney, 2007; Srinivasan, 2001; Muggleton, 1995; Quinlan & Cameron-Jones, 1993) that learn first-order rules from data assume that the examples are complete and correct and they cannot handle uncertainty in examples. However, in the machine reading task, these assumptions do not hold as facts that are implicit in the natural language text are never extracted and errors in the extraction process result in noisy, uncertain extractions. Further, most methods require a complete set of both positive and negative examples. But we have access to only positive examples and the closed world assumption cannot be applied here to generate negative examples. As a result, we cannot use existing methods to induce rules effectively. Instead, we propose to develop an algorithm that can learn the structure of BLPs from incomplete and uncertain data.

In the longer–term, we propose to extend our work in the following directions:

- *Multiple Predicate Learning*
  Existing methods for structure learning almost always learn first-order rules for a single target relation at a time since learning multiple predicates is a hard problem (De Raedt, Lavrac, & Dzeroski, 1993). Even in our proposed approach for learning first-order rules, we attempt to learn rules for a single

predicate at a time. We believe that we would be able to learn a better structure if we learned rules for all target predicates at the same time. In the longer-term, we propose to extend our approach to multiple predicate learning.

- *Discriminative parameter learning for BLPs*
  Kersting and De Raedt (2008) propose two different algorithms for learning the parameters of BLP from data – Expectation Maximization (EM) and gradient ascent based learning. Both methods learn parameters by optimizing the data log-likelihood, which makes them unsuitable for tasks that involve classification. We propose to explore approaches to discriminatively learn the parameters for BLPs so that the learned model would be better suited for classification tasks. To the best of our knowledge, no algorithm that learns the parameters discriminatively for BLPs has been developed to date.

- *Comparison of BALPs to other probabilistic logics on the plan recognition task*
  In this proposal, we compare the performance of BALPs to that of MLNs on the plan recognition task. There are other probabilistic logics like Abductive Stochastic Logic Programs (ASLPs) (Tamaddoni-Nezhad, Chaleil, Kakas, & Muggleton, 2006), PRISM (Sato, 1995), Poole's Horn Abduction (Poole, 1993) that perform abductive reasoning. However, these methods have not been applied to abductive plan recognition. We propose to compare the performance of BALPs with these systems in the future.

2 Background and Related Work

2.1 First-order logic

First-order logic is a formal language for representing relational domains involving several objects, their properties, and their relationships with other objects (Russell & Norvig, 2003). A *term* in first-order logic is a symbol that represents an object or an entity in the domain and every object in the domain has an associated type. There are three types of terms – constants, variables, and functions. A *constant* is a term that represents an individual object or an entity, while a *variable* is a term that acts as a template or a placeholder for a set of entities of the same type. A *function symbol*, represented by f/n is a term that represents a function over a set of terms; f is the name of the function symbol, and n is the arity, or the number of arguments/terms it takes. All constants are represented using strings that start with a lower–case letter (e.g mary, bob, alice), while all variables are represented using strings that start with an upper–case (e.g X1, Y1, Z1). A *predicate*, denoted by p/n represents a relation between entities in the domain; p is the name of the predicate, and n is its arity, the number of arguments/terms the predicate takes.

A literal is a predicate applied to terms. A positive literal is called an atom, and a negative literal is a negated atom. A literal that contains only constants is called a ground literal. A ground atom whose truth values is known is called a *fact*. A clause is an expression of the form

$$b_1 \bigwedge b_2 \bigwedge .... \bigwedge b_n \rightarrow h_1 \bigvee h_2 \bigvee ...... h_n$$

where '$\bigwedge$' represents a conjunction, '$\bigvee$' represents a disjunction, and '$\rightarrow$' stands for an implication. $b_1 \bigwedge b_2 \bigwedge .... \bigwedge b_n$ is called the *body* of the clause, while $h_1 \bigvee h_2 \bigvee ...... h_n$ is called the *head* of the clause. The above clause can also be written in the disjunctive normal form (DNF) as a disjunction of literals as follows:

$$\neg b_1 \bigvee \neg b_2 \bigvee .... \bigvee \neg b_n \bigvee h_1 \bigvee h_2 \bigvee ...... h_n$$

A Horn clause is a clause in DNF that contains at most one positive literal and a definite clause is one that contains exactly one positive literal. If b_i and h are atoms, then a definite clause has the form

$$b_1 \bigwedge b_2 \bigwedge .... \bigwedge b_n \rightarrow h$$

Given a logical formula, the variables in the formula are either universally quantified or existentially quantified. A variable is said to be universally quantified if it is true for all in the domain. On the other hand, a variable is said to be existentially quantified if it is true for some object in the domain. The symbol '$\forall$' is used for universal quantification, while the symbol '$\exists$' is used for existential quantification.

A substitution $\theta = \{V_1/t_1, V_2/t_2, ...V_n/t_n\}$ is an assignment of terms t_i to corresponding variables V_i. Given a formula (term, atom, clause) f and a substitution θ, then the instantiation $f\theta$ represents the formula obtained by replacing each variable V_i in the formula by its corresponding term t_i in θ. *Unification* is the process that takes two atomic sentences p and q and returns a substitution θ such that $p\theta = q\theta$, if both of them match, otherwise it returns failure.

Given a logic program, i.e a set of first-order clauses, the *Herbrand universe* is defined as the set of all ground terms that can constructed from the constants and the function symbols that are present in the program. For a function free logic program, the Herbrand universe reduces to the set of constants that occur in the clauses. The *Herbrand base* is the set of ground atoms over the Herbrand universe. The *Herbrand interpretation* is the set of ground atoms from the Herbrand base that are true. A Herbrand interpretation I is a model for a clause c if and only if for all substitutions θ such that $body(c)\theta \subset I \rightarrow head(c)\theta \in I$. I is a model for a set of clauses B if it is a model for every clause in B.

Automated inference in first-order logic involves using techniques like forward chaining and backward chaining. Given a knowledge base (KB) consisting of a set of formulae in first-order logic and a set of facts, forward chaining adds new facts to the knowledge base. For every implication $p \rightarrow q$ in the KB, if p is satisfied, i.e if p is true, then q is added to the KB, if it is not already present. On the other hand, given a KB and a query literal, backward chaining searches for those implications which can derive the query literal. For a query literal q, if an implication $p \rightarrow q$ is present, and if p is true, then backward chaining concludes that q is true, otherwise it will try to prove p. If p cannot be proved, then it fails to conclude q. *SLD resolution* is a backward chaining procedure for definite clauses.

2.2 Logical Abduction

Abduction, also called abductive reasoning, is defined as the process of finding the best explanation for a set of observations (Peirce, 1958). It is widely used in tasks such as plan/activity recognition and diagnosis that require inferring cause from effect (Ng & Mooney, 1992). Most previous approaches to abduction have been based on first-order logic and determine a small set of assumptions sufficient to deduce the observations (Pople, 1973; Levesque, 1989; Kakas, Kowalski, & Toni, 1993). In the logical framework, abduction, is usually defined as follows (Pople, 1973):

- **Given:** Background knowledge B and observations O, both represented as sets of formulae in first-order logic, where B is typically restricted to a set of Horn clauses and O is restricted to a conjunction of ground literals.
- **Find:** A hypothesis H, also a set of logical formulae, such that $B \cup H \not\models \bot$ and $B \cup H \models O$.

Here $\models$ means logical entailment and $\bot$ means false, i.e. find a set of assumptions that is consistent with the background theory and explains the observations. There are generally many hypotheses H that explain a particular set of observations O. The best hypothesis is typically selected based on the size (simplicity) of H, following Occam's Razor.

2.3 Inductive Logic Programming

Inductive logic programming (ILP) has been defined as the intersection of machine learning and logic programming (Muggleton, 1992). Given background knowledge B and a set of positive and negative examples for a target relation/predicate, ILP involves finding a hypothesis H, usually a definite logic program such that H along with B covers all positive examples, but none of the negative examples. The background knowledge B can either be a definite logic program or a set of ground literals that represent entities and relations in the domain. DeRaedt and Kersting (2004) discuss the following settings for ILP systems :

- Learning from entailment:
  In this setting, the induced hypothesis H, along with the background knowledge B entails all positive examples, but do not entail any of the negative examples. FOIL (Quinlan & Cameron-Jones, 1993), ALEPH (Srinivasan, 2001), PROGOL (Muggleton, 1995) are some of the ILP systems that are learn from entailment.

- Learning from interpretations:
  In this setting, examples represent Herbrand interpretations and the induced hypothesis H is said to cover an example if and only if the example is a Herbrand model of $B \bigcup H$, where B is the background knowledge base. CLAUDIEN (De Raedt & Dehaspe, 1997) is an ILP system that learns from interpretations. Learning from interpretations is easier than learning from entailment as the examples in the former are complete. Further, this setting is suitable for learning in domains where only positive examples are available.

- Learning from proofs:
  In this setting, examples are ground proof-trees and the induced hypothesis H is said to cover a example, if and only of the example is a proof of $B \bigcup H$, where B is the background knowledge base. Model Inference System (MIS) (Shapiro, 1983) is a system that learns from proofs.

2.4 Bayesian Networks

A Bayesian network is a directed acyclic graph that represents the joint probability distribution of a set of random variables in a compact manner (Koller & Friedman, 2009). Each node in the network represents a random variable and the directed edges between nodes represent the conditional dependencies between the random variables. If there is a directed edge from node a to node b, then the random variable represented by node b is conditionally dependent on that represented by node a. Absence of edges between nodes indicate conditional independence between the random variables. In a discrete Bayesian network, each node takes a discrete set of values. Associated with each node is a conditional probability table (CPT), which gives the probability of the node taking a certain value for different combination of values that the parent nodes take. The joint probability distribution for a Bayesian network is given by the following formula:

$$\mathrm{P}(X) = \prod_i P(X_i | Pa(X_i)),$$

where $X = X_1, X_2, ..., X_n$ represents the set of random variables in the network and $Pa(X_i)$ represents the parents of X_i. For the rest of this proposal, we will discuss only about discrete Bayesian networks.

2.4.1 Probabilistic Inference in Bayesian Networks

Inference in Bayesian networks generally involves computing the posterior probability of a query given the evidence. Koller and Friedman (2009) describe several algorithms to perform both exact and approximate inference in discrete Bayesian networks. For approximate inference, several sampling algorithms like forward sampling, likelihood weighting, Gibbs sampling etc. can be used. These algorithms generate a large number of random samples for the given Bayesian network, and then use these samples to compute the posterior probabilities. However, if the Bayesian network contains several deterministic constraints, i.e 0 values in the CPTs, then these sampling algorithms fail to generate sufficient samples, thus leading to a poor approximation of the posterior probability. In such cases, SampleSearch (Gogate & Dechter, 2007), an approximate sampling algorithm specifically designed for graphical models with multiple deterministic constraints can be used. The other type of inference in Bayesian networks involves computing the most probable explanation (MPE) (Pearl, 1988), which determines the joint assignment of values to the unobserved nodes in the network that has the maximum posterior probability given the evidence. It is possible to compute multiple alternative explanations using the k-MPE algorithm (Nilsson, 1998). There are several Bayesian network packages like Netica [1] and ELVIRA (Elvira-Consortium, 2002) that provide implementation for some of these algorithms. We use Netica for exact inference (joint and marginal), ELVIRA for MPE and k-MPE inference, and SampleSearch for approximate inference.

2.4.2 Learning Bayesian Networks

Learning Bayesian networks automatically from data involves learning the structure, i.e the conditional dependencies between the random variables, and learning the parameters, i.e the entries in the CPTs. Given a Bayesian network with a fixed structure, it is possible to learn the parameters automatically from data. When the data is fully observable, frequency counting is used to estimate the maximum likelihood (ML) parameters (Koller & Friedman, 2009). For partially observed data, Lauritzen (1995) has proposed an algorithm based on Expectation Maximization (EM), which maximizes the likelihood of the data. Russell et al. (1995) have proposed a gradient-ascent based parameter learning algorithm that also optimizes the likelihood of the data. On the other hand, Greiner and Zhou (2002) have developed a method for discriminatively learning the parameters by optimizing the conditional likelihood. Several methods have been proposed to learn the structure of Bayesian networks automatically from data (Koller & Friedman, 2009). Some methods use dynamic programming and its extensions to learn the structure (Koivisto & Sood, 2004; Silander & Myllymäki, 2006). Other methods learn the structure approximately by searching through the space of possible network structures (Heckerman & Chickering, 1995) or searching through the space of possible network orderings (Teyssier & Koller, 2005).

2.4.3 Noisy-Or and Noisy-And models

The noisy-or and noisy-and models (Pearl, 1988) are used to encode the CPTs for boolean nodes compactly using fewer parameters. In a discrete Bayesian network, the number of entries in the CPTs for any node is exponential in the number of parents. However, the noisy-or and noisy-and models require parameters that are linear in the number of parents for encoding the CPT. As a result, these models help reduce the number of parameters that have to estimated from data.

The *noisy-or model* is used when there are several different causes c_i for an event e and each cause independently triggers the event with a certain probability p_i. Let $leak$ be the probability that the event e

[1] `http://www.norsys.com/`

occurs due to an unknown cause. Then the probability of occurrence of e using the noisy-or is given as below:

$$P(e) = 1 - [(1 - leak)] \prod_i (c_i?(1 - p_i) : 1)$$

The notation $(c_i?(1 - p_i) : 1)$ means that if c_i is true, then the probability is $1 - p_i$, else it is 1.

The *noisy-and* model is used where there are several events c_i that have to occur simultaneously for the event e to occur and each event c_i fails to trigger e independently with a probability p_i. Let *inh* be the probability that e does not occur even when all events c_i have occurred. *inh* accounts for unknown events due to which e has failed to trigger. Then the probability of occurrence of e using the noisy-and model is as below:

$$P(e) = (1 - inh) \prod_i (c_i?1 : (1 - p_i))$$

The notation $(c_i?1 : (1 - p_i))$ means that if c_i is true, then the probability is 1, otherwise, it is $1 - p_i$

2.5 Bayesian Logic Programs

Bayesian logic programs (BLPs) (Kersting & De Raedt, 2001, 2007) can be considered as templates for constructing *directed* graphical models (Bayes nets). Given a knowledge base as a special kind of logic program, standard logical inference (SLD resolution) is used to automatically construct a Bayes net for a given problem. More specifically, given a set of facts and a query, all possible Horn-clause proofs of the query are constructed and used to build a Bayes net for answering the query. Standard probabilistic inference techniques described in Section 2.4.1 are then used to compute the most probable answer.

More formally, a BLP consists of a set of *Bayesian clauses*, definite clauses of the form $a|a_1, a_2, a_3, ...a_n$, where $n \geq 0$ and a, a_1, a_2, a_3,......,a_n are *Bayesian predicates* (defined below). a is called the head of the clause (head(c)) and (a_1, a_2, a_3,....,a_n) is the body (body(c)). When $n = 0$, a Bayesian clause is a fact. Each Bayesian clause c is assumed to be universally quantified and range restricted, i.e $variables\{head\} \subseteq variables\{body\}$, and has an associated *conditional probability distribution* cpd(c) = P(head(c)|body(c)).

A *Bayesian predicate* is a predicate with a finite domain, and each ground atom for a Bayesian predicate represents a random variable. Associated with each Bayesian predicate is a combining rule such as *noisy-or* or *noisy-and* that maps a finite set of cpds into a single cpd (cf. Section 2.4.3). Let a be a Bayesian predicate defined by two Bayesian clauses, $a|a_1, a_2, a_3, ...a_n$ and $a|b_1, b_2, b_3, ...b_n$, where cpd_1 and cpd_2 are their cpd's. Let θ be a substitution that satisfies both clauses. Then, in the constructed Bayes net, directed edges are added from the nodes for each $a_i\theta$ and $b_i\theta$ to the node for $a\theta$. The combining rule for a is used to construct a single cpd for $a\theta$ from cpd_1 and cpd_2. The probability of a joint assignment of truth values to the final set of ground propositions is then defined in the standard way for a Bayes net:

$$\text{P}(X) = \prod_i P(X_i|Pa(X_i)),$$

where $X = X_1, X_2, ..., X_n$ represents the set of random variables in the network and $Pa(X_i)$ represents the parents of X_i. Once a ground network is constructed, standard probabilistic inference methods can be used to answer various types of queries as described in Section 2.4.1.

2.5.1 Learning Bayesian Logic Programs

Learning a BLP from data involves learning the structure, the set of first-order Horn clauses and the parameters – the cpd entries and the parameters for the combining rules. Given a BLP with a fixed structure, the parameters of the BLP can be learned automatically from data using the methods proposed by Kersting and DeRaedt (2008). In their first method, Kersting and DeRaedt have adapted the Expectation Maximization (EM) algorithm developed for propositional Bayesian networks by Lauritzen (1995), and in their second method, they have adapted the gradient-ascent based algorithm developed by Russell et al. (1995) for Bayesian networks. Note that both the algorithms proposed by Kersting and DeRaedt for learning the parameters of a BLP optimize the likelihood of the data. Kersting and DeRaedt have also proposed an algorithm to learn the structure of the BLP based on the model of learning from interpretations (Kersting & Raedt, 2008). In their method, each example in the training data represents the least Herbrand model of the target BLP. The algorithm performs a hill climbing search through the space of possible structures by optimizing the likelihood of the data. They use CLAUDIEN (De Raedt & Dehaspe, 1997) to get the initial set of structures for the search.

3 Extending Bayesian Logic Programs for Abductive Plan Recognition

In this section, we discuss our completed work in which we have extended BLPs to the task of abductive plan recognition. Plan recognition is the task of predicting an agent's top-level plans based on its observed actions. It is an abductive reasoning task that involves inferring cause from effect (Charniak & McDermott, 1985). It is used in several applications like story understanding, strategic planning, intelligent user interfaces, etc. In story understanding, the character's motives or plans have to be recognized based on its actions in order to answer questions about the story. In strategic planning system where there are several agents performing several actions, it becomes necessary for each agent to recognize plans of other agents so that they can work cooperatively. In an intelligent user interface, plan recognition is used to predict the task the user is performing so that the system could give valuable tips to the user to perform the task more efficiently.

Traditionally, plan-recognition approaches have been based on first-order logic in which a knowledge-base of plans and actions is developed for the domain and then default reasoning (Kautz & Allen, 1986) or logical abduction (Ng & Mooney, 1992) is used to predict the best plan based on the observed actions. However, these approaches are unable to handle uncertainty in the observations or background knowledge and are incapable of estimating the likelihood of different plans. An alternative approach to plan recognition is to use probabilistic methods such as Abstract Hidden Markov Models (Bui, 2003), probabilistic context-free grammars (Pynadath & Wellman, 2000), Bayesian networks (Charniak & Goldman, 1989, 1991; Huber, Durfee, & Wellman, 1994; Horvitz & Paek, 1999), or statistical *n*-gram models (Blaylock & Allen, 2005b). While these approaches handle uncertainty, they cannot handle structured representations as they are essentially propositional in nature. As a result, it is also difficult to incorporate planning domain knowledge in these approaches.

As mentioned earlier, the last few years have seen a development of several SRL formalisms like MLNs and BLPs. Of these formalisms, MLNs have been applied to abductive plan recognition by Kate and Mooney (2009). Since MLNs employ deduction for logical inference, they adapt MLNs for abduction by adding reverse implications for every rule in the knowledge base. However, the addition of these rules increases the size and complexity of the MLN, resulting in a computationally expensive model. We refer to this MLN model by Kate and Mooney as MLN-PC. In order to overcome the limitations of MLN-PC, Singla and

Mooney (2011) have developed two new approaches to plan recognition using MLNs. In the first approach, they extend MLN-PC by adding hidden causes, which reduces the complexity of ground networks. In the second approach, they introduce a novel procedure to construct the ground networks using logical abduction. We refer to Singla and Mooney's first approach as MLN-HC and the second one as MLN-HCAM. While MLN-HC performs slightly better than MLN-PC, it still does not scale to large domains. MLN-HCAM outperforms both MLN-HC and MLN-PC.

In this proposal, we apply Bayesian Logic Programs (BLPs), to abductive plan recognition. Since it is an abductive reasoning task that involves inferring cause from effect, we believe that a directed probabilistic logic like BLPs will be better suited for plan recognition than an undirected probabilistic logic like MLNs. Further, since BLPs include only those rules that are relevant to the query in the ground network, we hypothesize that BLPs can overcome some of the limitations of MLN-PC (Kate & Mooney, 2009) and MLN-HC (Singla & Mooney, 2011) with respect to scaling to large domains. However, BLPs use SLD resolution to generate proof trees, which are then used to construct a ground Bayes net for a given query. Since SLD resolution is a deductive inference, it is unable to construct proofs for abductive problems such as plan recognition because deductive inference involves predicting effects from causes, while the plan recognition task involves inferring causes (top-level plans) from effects (observations). Therefore, we extend BLPs to use logical abduction to construct proofs. In logical abduction, missing facts are assumed when necessary to complete proof trees, and we use the resulting abductive proof trees to construct Bayes nets. We call the resulting model Bayesian Abductive Logic Programs (BALPs). Like all SRL formalisms, BALPs combine the strengths of both first-order logic and probabilistic graphical models, thereby overcoming the limitations of traditional plan recognition approaches mentioned above.

We first describe the abductive inference procedure used in BALPs. Next we describe how probabilistic parameters are specified and how probabilistic inference is performed. Then, we discuss how parameters can be automatically learned from data. Finally, we apply BALPs to the task of plan recognition and demonstrate its efficacy on three benchmark data sets from three different application domains – story understanding, strategic planning, and intelligent user interfaces. We also compare BALP's performance with several other plan-recognition methods.

3.1 Abdutive Inference in BALPs

Let O_1, O_2, ..., O_n be the set of observations. We derive a set of most-specific abductive proof trees for these observations using the method originally proposed by Stickel (1988). The abductive proofs for each observation literal are computed by backchaining on each O_i until every literal in the proof is proven or assumed. A literal is said to be proven if it unifies with some fact or the head of some rule in the knowledge base, otherwise it is said to be assumed. Since multiple plans/actions could generate the same observation, an observation literal could unify with the head of multiple rules in the knowledge base. For such a literal, we compute alternative abductive proofs. The resulting abductive proof trees are then used to build the structure of the Bayes net using the standard approach for BLPs.

The basic algorithm to construct abductive proofs is given in Algorithm 1. The algorithm takes as input a knowledge base (KB) in the form of Horn clauses and a set of observations as ground facts. It outputs a set of abductive proof trees by performing logical abduction on the observations. These proof trees are then used to construct the Bayesian network. For each observation O_i, AbductionBALP searches for rules whose consequents unify with O_i. For each such rule, it computes the substitution from the unification process and substitutes variables in the body of the rule with bindings from the substitution. The literals in the body now become new subgoals in the inference process. If these new subgoals cannot be proved, i.e if they cannot unify with existing facts or with the consequent of any rule in the KB, then they are assumed. In order to

Algorithm 1 AbductionBALP

Inputs: Background knowledge KB and observations O_1, O_2, O_3, ..., O_n both represented as sets of formulae in first-order logic, where KB is typically restricted to a set of Horn clauses and each O_i is a ground literal.

Output: Abductive proofs for all O_i.

```
 1: Let $Q$ be a queue of unproven atoms, initialized with $O_i$
 2: while $Q$ not empty do
 3:   $A_i$ ← Remove atom from $Q$
 4:   for each rule $R_i$ in $KB$ do
 5:     $consequent$ ← Head literal of $R_i$
 6:     if $A_i$ unifies with $consequent$ then
 7:       $S_i$ ← unify $A_i$ and $consequent$ and return substitution
 8:       Replace variables in the body of $R_i$ with bindings in $S_i$. Each literal in the body of $R_i$ is a new
          subgoal.
 9:       for each $literal_i$ in body of $R_i$ do
10:         if $literal_i$ unifies with head of some rule $R_j$ in $KB$ then
11:           add $literal_i$ to $Q$
12:         else if $literal_i$ unifies with an existing fact then
13:           Unify and consider the literal to be proved
14:         else
15:           if $literal_i$ unifies with an existing assumption then
16:             Unify and update the assumption
17:           else
18:             Assume $literal_i$ by replacing any unbound variables that are existentially quantified in
                $literal_i$ with new Skolem constants.
19:           end if
20:         end if
21:       end for
22:     end if
23:   end for
24: end while
```

(a)
# Shopping
1. inst(G,going) | inst(B,shopping), go-step(B,G).
2. inst(SP,shopping-place) | inst(S,shopping), store(S,SP).
# Robbing
3. inst(P,going) | inst(R,robbing), go-step(R,P).

(b)
inst(go1,going)
inst(store1,shopping-place)

(c)
inst(go1,going) | inst(a1,shopping), go-step(a1,go1).
inst(go1,going) | inst(a1,robbing), go-step(a1,go1).
inst(store1,shopping-place) | inst(a1,shopping), store(a1,store1).

Figure 1: (a) A partial knowledge base from the Story Understanding data set. (b) The logical representation of the observations. (c) The set of ground rules obtained from logical abduction.

minimize the number of assumptions, the assumed literals are first matched with existing assumptions. If no such assumption exists, then any unbound variables in the literal that are existentially quantified are replaced by Skolem constants.

In SLD resolution, which is used in BLPs, if any subgoal literal cannot be proven, the proof fails. However, in BALPs, we assume such literals and allow proofs to proceed till completion. Note that there could be multiple existing assumptions that could unify with subgoals in Step 15. However, if we used all ground assumptions that could unify with a literal, then the size of the ground network would grow exponentially, making probabilistic inference intractable. In order to limit the size of the ground network, we unify subgoals with assumptions in a greedy manner. We found that this approach worked well for the plan-recognition domains we explored. For other tasks, domain-specific heuristics could potentially be used to reduce the size of the network.

We now illustrate the abductive inference process with a simple example from the Story-Understanding benchmark data set described in Section 3.4.1. Consider the partial knowledge base and set of observations given in Figure 1a and Figure 1b respectively. There are two top-level plans, shopping and robbing, in the knowledge base. Note that the action literal “inst(G, going)” could be observed as part of both shopping and robbing. For each observation literal in Figure 1b, we recursively backchain to generate abductive proof trees. When we backchain on the literal *inst(go1,going)* using Rule 1, we obtain the subgoals *inst(B,shopping)* and *go-step(B,go1)*. These subgoals become assumptions since no observations or heads of clauses unify with them. Since B is an existentially quantified variable, we replace it with a Skolem constant $a1$ to obtain the ground assumptions *inst(a1,shopping)* and *go-step(a1,go1)*. We then backchain on literal *inst(go1,going)* using Rule 3 to get subgoals *inst(R,robbing)* and *go-step(R,go1)*. We cannot unify *inst(R, robbing)* with any observation or existing assumptions; however, we can unify *go-step(R,go1)* with an existing assumption *go-step(a1,go1)*, thereby binding R to a1. In order to minimize the number of assumptions, we first try to match literals with unbound variables to existing assumptions, rather than instantiating them with new Skolem constants. Finally, we backchain on the literal *inst(store1,shopping-place)* using Rule 2 to get subgoals *inst(S,shopping)*, *store(S,store1)*. Here again, we match *inst(S, shopping)* to an existing

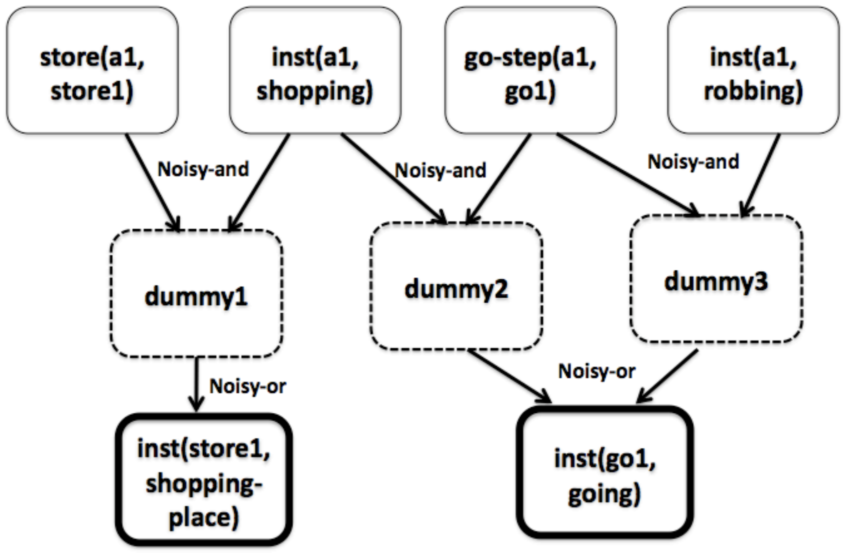

Figure 2: Bayesian network constructed for example in Figure 1. The nodes with thick borders represent observed actions, the nodes with dotted borders represent intermediate nodes used to combine the conjuncts in the body of a clause, and the nodes with thin borders represent plan literals.

assumption *inst(a1,shopping)*, thereby binding *S* to a1.

Figure 1c gives the final set of ground rules generated by abductive inference. After generating all abductive proofs for all observation literals, we construct a Bayesian network. Figure 2 shows the Bayesian network constructed for the example in Figure 1. Note that since there are no observations/facts that unify with the subgoals (*inst(B,shopping)*, *go-step(B,G)*, *inst(R,robbing)*, *go-step(R,P)*, and *store(S,SP)*) generated during backchaining on observations, SLD resolution will fail to generate proofs. This is typical in plan recognition, and as a result, we cannot use BLPs for such tasks.

The only difference between BALPs and BLPs lies in the logical inference procedure used to construct proofs. Once the abductive proofs are generated, BALPs use the same procedure as BLPs to construct the Bayesian network. We further show in Section 3.3 and Section 3.4.3 that techniques developed for BLPs for learning parameters can also be used for BALPs.

3.2 Probabilistic Parameters and Inference

We now discuss how parameters are specified in BALPs. We use noisy/logical-and and noisy-or models to specify the *cpd*s in the ground Bayesian network as these models compactly encode the *cpd* with fewer parameters, i.e. just one parameter for each parent node. Depending on the domain, we use either a strict *logical-and* or a softer *noisy-and* model to specify the *cpd* for combining evidence from the conjuncts in the body of a clause. We use a noisy-or model to specify the *cpd* for combining the disjunctive contributions from different ground clauses with the same head. Figure 2 shows the noisy-and and noisy-or nodes in the Bayesian network constructed for the example in Figure 1.

Given the constructed Bayesian network and a set of observations, we determine the best explanation using standard methods for computing the Most Probable Explanation (MPE) (cf. Section 2.4.1), which determines the joint assignment of values to the unobserved nodes in the network that has the maximum posterior probability given the observations. To compute multiple alternative explanations, we use the k-

MPE algorithm (Nilsson, 1998) as implemented in Elvira (Elvira-Consortium, 2002). For other types of exact probabilistic inference (marginal and joint) we use Netica,[2] a commercial Bayes-net software package.

When the complexity of the ground network makes exact inference intractable (as in the Monroe dataset described in Sect. 3.4), we have to resort to approximate inference. Due to the (noisy/logical) *and* and *or* nodes in the network, there are a number of deterministic constraints, i.e. 0 values in the *cpd*s. As a result, generic importance sampling algorithms like likelihood weighting failed to generate sufficient samples. Hence, we used SampleSearch (Gogate & Dechter, 2007), an approximate sampling algorithm specifically designed for graphical models with multiple deterministic constraints.

3.3 Parameter Learning

Learning can be used to automatically set the noisy-or and noisy-and parameters in the model. We learn these parameters using the EM algorithm adapted for BLPs by Kersting and De Raedt (2008). In supervised training data for plan recognition, one typically has evidence for the observed actions and the top-level plans. However, we usually do not have evidence for network nodes corresponding to subgoals, noisy-ors, and noisy/logical-ands. As a result, there are a number of variables in the ground networks which are always hidden, and hence EM is appropriate for learning the requisite parameters from the partially observed training data. We simplify the problem by learning only the noisy-or parameters and using a deterministic logical-and model to combine evidence from the conjuncts in the body of a clause. We use uniform priors for top-level plans unless otherwise mentioned.

3.4 Experimental Evaluation

In this section, we evaluate BALPs on three plan-recognition datasets. Unfortunately, there are very few benchmark datasets or rigorous experimental evaluations of plan recognition. First, we describe experiments to demonstrate that BALPs are more effective for plan recognition than previous approaches. Then, we describe additional experiments to demonstrate that the EM algorithm can learn parameters of the BALP model effectively.

3.4.1 Datasets

- Monroe / Reformulated Monroe

  We used the Monroe dataset, an artificially-generated plan-recognition dataset in the emergency response domain by Blaylock and Allen (2005a). This domain includes top level plans such as setting up a temporary shelter, clearing a road wreck, and providing medical attention to victims. The task is to infer a single top level plan from a set of observed actions automatically generated by a planner. The planner used to construct plans is SHOP2 (Nau, Ilghami, Kuter, Murdock, Wu, & Yaman, 2003) and the domain knowledge is represented as a hierarchical transition network (HTN). We constructed a logical knowledge base representing the domain knowledge encoded in the HTN. We used 1,000 artificially generated examples in our experiments. Each example instantiates one of 10 top-level plans and contains an average of 10.19 literals describing a sample execution of this plan. This data set is an example of plan recognition being applied to strategic planning.

  Due to computational complexity, we were unable to compare the performance of BALPs with Kate and Mooney's MLN-PC (2009) approach on this domain. Their approach resulted in an MLN with

[2]http://www.norsys.com/

rules containing multiple existentially quantified variables which produced an exponential number of possible groundings, eventually leading to memory overflow. In order to compare BALPs with MLN-PC, we slightly modified the Monroe domain to eliminate this problem without significantly changing the underlying task. The resulting dataset also had 1,000 examples, with an average of 9.7 observations per example. We refer to this dataset as "Reformulated-Monroe."

- Linux

  The Linux dataset is another plan-recognition dataset created by Blaylock and Allen (2004). Human users were asked to perform various tasks in Linux and their commands were recorded. The task is to predict the correct top level plan from the sequence of executed commands. For example, one of the tasks involves finding all files with a given extension. The dataset consists of 19 top level plans and 457 examples, with an average of 6.1 command literals per example. We constructed the background knowledge base for the Linux dataset based on our knowledge of the commands. This data set is an example of plan recognition being applied to intelligent user interfaces.

- Story Understanding

  We also used a dataset[3] that was previously used to evaluate abductive story understanding systems (Ng & Mooney, 1992; Charniak & Goldman, 1991). In this task, characters' higher-level plans must be inferred from their actions described in a narrative text. A logical representation of the literal meaning of the text is given for each example. A sample story is: "Bill went to the liquor-store. He pointed a gun at the owner." The plans in this dataset include shopping, robbing, restaurant dining, traveling in a vehicle (bus, taxi or plane), partying and jogging. Some narratives involve more than a single plan. This small dataset consists of 25 development examples and 25 test examples each containing an average of 12.6 literals. We used the background knowledge that was initially constructed for the ACCEL system (Ng & Mooney, 1992). This data set is an example of plan recognition being applied to story understanding. Figure 1a and Figure 1b give a sample knowledge base and a set of observations from the Story Understanding data set.

Apart from the fact that each data set comes from a different application domain, all data sets described above test certain specific aspects of the plan recognition system. Since the Monroe domain is very large with several subgoals and entities, it tests the ability of the plan recognition system to scale to large domains. On the other hand, the Linux data set does not have a large domain. However, since the data is from human users, it is noisy. There are several sources of noise including the case in which the human users have reported that they have successfully executed the top-level plan even though they have not (Blaylock & Allen, 2005b). As a result, this data set tests the robustness of the plan recognition system. The plan recognition task on Monroe and Linux domains involve prediction of a *single* top-level plan based on the observed actions. However, on the Story Understanding domain, most examples have multiple top-level plans as the answer. This data set tests the ability of a plan recognition system to identify all possible top-level plans based on the observed actions.

3.4.2 Comparison with Other Approaches

We now present comparisons to previous approaches to plan-recognition across different benchmark datasets.

[3] `http://www.cs.utexas.edu/~ml/accel.html`

- Monroe and Linux

  We first compared BALPs with MLN-HCAM (Singla & Mooney, 2011) and Blaylock and Allen's (2005b) plan-recognition system on both the Monroe and Linux datasets. Blaylock and Allen's approach learns statistical n-gram models to separately predict plan schemas (i.e. predicates) and their arguments. We were unable to run MLN-PC and MLN-HC on these domains due to scaling issues.

  We learn the noisy-or parameters for BALPs using the EM algorithm described in Sect. 3.3 for both Linux and Monroe domains. We initially set all noisy-or parameters to 0.9 and this gave reasonable performance in both domains. We ran EM with several different starting points including random weights and manual weights (0.9). We found that running EM starting with manual weights generally performed the best for both domains, and hence we used the weights learned from this model for comparison.

  For Linux, we performed 10-fold cross validation for evaluation and we ran EM till convergence on the training set for each fold. For Monroe, where more data is available, we used 300 examples for training, 200 examples for validation, and the remaining 500 examples for testing. We ran EM iterations on the training set until the accuracy on the validation set stopped improving. We then used the final learned set of weights to perform plan-recognition on the test set.

  For both Monroe and Linux, the plan-recognition task involves inferring a *single* top level plan that best explains the observations. Hence, we computed the marginal probabilities for all ground instantiations of the plan predicates in the network and picked the single plan instantiation with the highest marginal probability.

  Due to differences in Blaylock and Allen's experimental methodology and ours, we are only able to directly compare performance using the convergence score (Blaylock & Allen, 2005b). The convergence score is the fraction of examples for which the correct plan predicate is inferred (ignoring the arguments) when given *all* of the observations.

  Table 1 shows the results. BALPs outperform both MLN-HCAM and Blaylock and Allen's system on the convergence score in both domains. An '*' indicates that the difference in the performance with respect to BALPs was statistically significant ($p < .05$) as determined by unpaired t-test[4]. The convergence scores for MLN-HCAM and Blaylock and Allen's system on Monroe are already quite high, leaving little room for improvement. However, BALPs was still able to improve over MLN-HCAM by 1.57% and Blaylock and Allen's system by 4.5%. On the other hand, the baseline convergence scores for Linux were fairly low, and BALPs were able to improve over MLN-HCAM by 13.89% and Blaylock and Allen's system by a remarkable 29.1%. Despite this improvement, the overall convergence score for Linux is not that high. Noise in the data is one reason for the modest score. Another issue with this data set is the presence of very similar plans, like find-file-by-ext and find-file-by-name. The commands executed by users in these two plans are nearly identical, making it difficult for a plan recognition system to distinguish them (Blaylock & Allen, 2004).

- Reformulated-Monroe

  We also compared the performance of BALPs with Kate and Mooney's (2009) MLN-PC approach on the Reformulated-Monroe dataset[5]. For MLN-PC, we were unable to effectively learn clause weights

[4]We did not have access to scores on individual examples for Blaylock and Allen's system, hence we performed the unpaired t-test.

[5]We were unable to compare to MLN-HC and MLN-HCAM (Singla & Mooney, 2011) as the results were not available on this data set.

	BALPs	MLN-HCAM	Blaylock and Allen
Convergence-Monroe	**98.4**	96.88	94.2*
Convergence-Linux	**46.6**	40.89*	36.1*

Table 1: Results for BALPs, MLN-HCAM, and the system by Blaylock and Allen on Monroe and Linux. '*' indicates that the difference is statistically significant wrt BALPs.

	Convergence Score	Accuracy-100	Accuracy-75	Accuracy-50	Accuracy-25
BALP	**99.90**	**97.40**	**66.80**	**32.67**	**9.83**
MLN-PC	79.66*	79.20*	40.51*	19.26*	4.10*

Table 2: Comparative Results for Reformulated-Monroe. "*" indicates that the differences in the performance are statistically significant.

on this dataset since it was intractable to run Alchemy's existing weight-learners due to the sizes of the MLN and data. Hence, we manually set the weights using the heuristics described by Kate and Mooney (2009). To ensure a fair comparison, we use manual weights for BALPs instead of using learned weights; we uniformly set all noisy-or parameters to .9 and used logical-and to combine the evidence from the conjuncts as this model gave reasonable performance on other domains as well.

We used two different performance measures to compare the performance of the two approaches – convergence score and accuracy. We compared the inferred plan with the correct plan to compute the accuracy score. When computing accuracy, partial credit was given for predicting the correct plan predicate with only a subset of its correct arguments. A point was rewarded for inferring the correct plan predicate, then, given the correct predicate, an additional point was rewarded for each correct argument. For example, if the correct plan was $plan_1(a_1, a_2)$ and the inferred plan was $plan_1(a_1, a_3)$, the accuracy was 66.67%.

The observation set for this domain includes all actions executed to achieve the top level plan. In order to evaluate performance for partially observed plans, we performed plan recognition given only subsets of the complete action sequence. Specifically, we report results after observing the first 25%, 50%, 75%, and 100% of the executed actions.

Table 2 shows the results. "Accuracy-*i*" is the accuracy when given the first *i*% of the observations. BALPs consistently outperform the MLN approach on this data set and the differences in their performance are statistically significant as determined by the Wilcoxon Sign Rank (WSR) test (Rosner, 2005). Significance was concluded at the 0.05 level. The convergence score for BALPs improved over that for MLNs by a significant 25.41%.

- Story Understanding

  On Story Understanding, we compared the performance of BALPs to MLN-HC (Singla & Mooney, 2011), MLN-HCAM (Singla & Mooney, 2011), MLN-PC (Kate & Mooney, 2009) and ACCEL (Ng & Mooney, 1992), a purely logic-based system that uses a metric to guide its search for selecting the best explanation. ACCEL can use two different metrics: *simplicity*, which selects the explanation with the fewest assumptions and *coherence*, which selects the explanation that maximally connects the input observations. This second metric is specifically geared towards text interpretation by measuring

	BALP	MLN-HCAM	MLN-HC	MLN-PC	ACCEL-Sim	ACCEL-Coh
Precision	72.07	69.13	67.08	67.31	66.45	89.39*
Recall	85.57	75.32*	78.94*	68.10*	52.32*	89.39
F-measure	78.24	72.10	72.53	67.70*	58.54*	89.39*

Table 3: Comparative Results for Story Understanding. "*" indicates that the differences wrt BALPs are statistically significant.

explanatory coherence (Ng & Mooney, 1990). Currently, this bias has not been incorporated in either the BALP or any of the MLN approaches.

For BALPs, we were unable to learn useful parameters from just 25 development examples. As a result, we set parameters manually trying to maximize performance on the development set. As before, a uniform value of 0.9 for all noisy-or parameters seemed to work well for this domain. Unlike other domains, using the logical-and model to combine the evidence from conjuncts in the body of the clause did not yield good results on Story Understanding. However, we found that using the noisy-and model improved the results by a fair margin; so we set the noisy-and parameters to a uniform value of .9. However, to disambiguate between conflicting plans, we set different priors for high level plans to maximize performance on the development data.

Since multiple plans are possible in this domain, we compared the inferred plans with the ground truth to compute *precision* , *recall*, and *F-measure*, the harmonic mean of precision and recall. As before, partial credit was given for predicting the correct plan predicate with some incorrect arguments. The observed literals in this data are already incomplete and do not include all of the actions needed to execute a plan, so they were used as is.

Table 3 shows the results. An "*" indicates that the difference in the performances of a given method and that of BALPs is statistically significant. BALPs performed better than ACCEL-Simplicity (ACCEL-Sim) and the different MLN based approaches. With respect to F-measure, BALPS improved over MLN-HCAM by 8.52%, MLN-HC by 7.87%, MLN-PC by 15.57%, and ACCEL-Simplicity by a significant 33.65%. However, ACCEL-Coherence (ACCEL-Coh) still performed the best. Since the coherence metric incorporates extra criteria specific to story understanding, this bias would need to be included in the probabilistic models to make them more competitive. However, the coherence metric is specific to narrative interpretation and not applicable to plan recognition in general.

Overall, we found that BALPs outperformed most existing approaches on the benchmark data sets, thus demonstrating that BALPs are effective for plan recognition.

3.4.3 Parameter Learning Experiments

We now describe additional experiments that we performed to understand the EM algorithm for learning the parameters for BALPs. These experiments are designed to demonstrate that the EM algorithm is effective for learning the parameters for BALPs on different plan recognition domains.

- Learning Methodology

	Convergence Score	Accuracy-100	Accuracy-75	Accuracy-50	Accuracy-25
MW	39.82	30.41	28.22	21.84	18.34
MW-Start	**46.6***	**36.32***	**34.06***	**25.45***	**19.83***
Rand-Start	41.57	31.4	29.1	20.53	14.55*

Table 4: Results for parameter learning on Linux. "*" indicates that the differences wrt MW model are statistically significant.

We used EM as described in Sect. 3.3 to learn noisy-or parameters for the Linux and Monroe domains.[6] We initially set all noisy-or parameters to 0.9. This gives reasonable performance in both domains, so we compare BALPs with learned noisy-or parameters to this default model which we call "Manual-Weights" (MW). For training, we ran EM with two sets of starting parameters – manual weights (0.9) and random parameters. We call the former "MW-Start" and the latter "Rand-Start". We used the same training and test splits as described above for Linux and Monroe domains. To measure performance, we computed convergence score and accuracy score for various levels of observability as described above.

- Learning Results

  Table 4 shows the results for different models on Linux. An "*" indicates that the difference in the performance scores for MW and the given model is statistically significant. MW-Start consistently outperforms MW, demonstrating that parameter learning improves the performance of default BALP parameters on the Linux domain. Rand-Start does marginally better than MW for all but 50% and 25% levels of partial observability. However, it does not perform as well as MW-Start, showing that learning from scratch is somewhat better than using default parameters but not as effective as starting learning from reasonable default values.

  Table 5 shows the results for different models on Monroe. The performance of MW is already so high that there is little room for improvement, at least with respect to the convergence score. As a result, the MW-Start model could not improve substantially over the MW model. The manual parameters seem to be at a (local) optimum, preventing EM from making further improvements on this data. Rand-Start is performing about as well, sometimes better and sometimes worse than MW, demonstrating that starting from random values the system can learn weights that are about as effective as manual weights for this domain. One reason for the high performance of the MW model on Monroe is the lack of ambiguity in the observations, i.e. there are few observed actions that are part of more than one possible plan. Overall, EM was able to automatically learn effective parameters for BALPs.

3.4.4 Discussion

We now discuss various aspects of BALPs that have led to its superior performance over existing methods. As mentioned earlier, MLN-PC (Kate & Mooney, 2009) and MLN-HC (Singla & Mooney, 2011) approaches cannot be applied to large domains like Monroe since the addition of reverse implications results in a computationally expensive model. As opposed to the explosive grounding of rules in MLN-PC and MLN-HC, BALPs use logical abduction in which only those rules that are relevant to the query are included in the

[6]We were unable to learn useful parameters for Story Understanding since the mere 25 development examples were insufficient for training.

	Convergence Score	Accuracy-100	Accuracy-75	Accuracy-50	Accuracy-25
MW	98.4	79.16	46.06	20.67	7.2
MW-Start	98.4	79.16	44.63*	20.26	7.33
Rand-Start	98.4	79.86*	44.73*	19.7*	10.46*

Table 5: Results for parameter learning on Monroe. "*" indicates that the differences wrt MW model are statistically significant.

ground network. This results in networks that are much smaller in size, thus enabling BALPs to scale to large domains. Like BALPs, MLN-HCAM (Singla & Mooney, 2011) also uses logical abduction to reduce the size of the ground networks. Instead of constructing ground Markov networks directly from the abductive proofs like in BALPs, MLN-HCAM supplies these proofs to the normal grounding process in MLNs in order to construct the ground networks. The abductive proofs from logical abduction help to limit the explosive grounding process in MLN-HCAM to a certain extent, but not completely. As a result, BALPs outperform MLN-HCAM as well. Further, the use of logical abduction allows BALPs to use an existing knowledge base that was created for planning without any modification.

When Blaylock and Allen (2005b) perform instantiated plan recognition, it is done in a pipeline of two separate steps. The first step predicts the plan schema and the second step predicts the arguments given the schema. Unlike their approach, BALPs are able to jointly predict both the plan and its arguments simultaneously. We believe that BALP's ability to perform joint prediction of plans and their arguments is at least partly responsible for its superior performance. In both BALP and MLN systems for plan recognition, the domain knowledge is encoded in the knowledge base, while the system by Blaylock and Allen has no access to the domain knowledge. We believe that the ability of BALPs to incorporate domain knowledge is also responsible for its superior performance.

Blaylock and Allen's system (2005b) uses 4500 examples to learn reasonable parameters on the Monroe domain. MLN-PC and MLN-HC approaches do not even scale on the Monroe domain. On the other hand, BALPs use only 300 examples for learning parameters on the Monroe domain, proving that the EM algorithm can effectively learn parameters when given reasonable number of examples. Except for the Story Understanding data set, the EM algorithm used in BALPs could learn parameters automatically from data. The inability of the EM algorithm to learn parameters on this data set could be attributed to the lack of sufficient examples more than anything. As per Kate and Mooney (2009), even the MLN-PC approach could not learn reasonable weights on the Story Understanding data set due to lack of sufficient examples. Note that it is possible to learn parameters for Reformulated-Monroe using EM, but we did not deliberately learn these parameters to ensure fair comparison with MLNs. Overall, the success of EM algorithm for learning parameters of BALPs on the original Monroe and Linux domains demonstrates that our approach allows for automatic learning of parameters from data. As a result, our approach does not require any manual setting or tuning of parameters.

In conclusion, we find that our approach to plan recognition using BALPs is very effective. Our results demonstrate that BALPs outperform most existing approaches on all benchmark data sets. As mentioned earlier, each data set in our evaluation tests a specific aspect of the system. BALP's superior performance on all benchmark data sets demonstrates that BALPs is a robust system for plan recognition.

3.5 Related Work

Some of the early work in plan recognition was done by Kautz and Allen (1986, 1987). They use deductive inference to predict plans using observed actions, an action taxonomy, and a set of commonsense rules or constraints. Lesh and Etzioni's approach (1995) to goal recognition constructs a graph of goals, actions, and their schemas and prunes the network until the plans present in the network are consistent with the observed goals. Both these approaches cannot disambiguate between competing plans.

There are several approaches to plan recognition using Bayesian networks (Charniak & Goldman, 1989, 1991; Huber et al., 1994). Based on the observed actions and a knowledge base constructed for planning, these approaches automatically construct Bayesian networks using different heuristics. Their work is similar to BALPs, but special purpose procedures are used to construct the necessary ground networks rather than using a general-purpose probabilistic predicate logic like MLNs or BLPs. Horvitz and Paek (1999) develop an approach that uses Bayesian networks to recognize goals in an automated conversation system; however their approach does not handle relational data.

Pynadath and Wellman (2000) extend probabilistic context-free grammars to plan recognition; Bui (2003) uses Abstract Hidden Markov Models for hierarchical goal recognition; Saria and Mahadevan (2004) extend work by Bui to multiagent plan recognition systems; Albrecht et al. (1998) develop an approach based on dynamic Bayesian networks to predict plans in an adventure game; however, none of these approaches can handle relational data. We have already discussed the other systems for plan recognition (Ng & Mooney, 1992; Kate & Mooney, 2009; Blaylock & Allen, 2005b) in Section 3.4.2.

Poole (1993) has developed a framework for Horn clause abduction using Bayesian networks. Chen *et. al* (2008) extend Stochastic Logic Programs (Muggleton, 2003) to incorporate abduction. Sato (1995) has also developed a probabilistic logic called PRISM that performs abduction. However, none of these approaches have been applied to plan recognition.

3.6 Summary

We introduced a new approach to plan recognition using Bayesian Logic Programs (BLPs) in this section. We extended BLPs for plan abductive plan recognition by employing logical abduction to construct the Bayesian networks as opposed to the standard logical deduction used in BLPs. We call the resulting model Bayesian Abductive Logic Programs (BALPs). We also demonstrated that the parameters of the BALP model can be learned automatically using the EM algorithm. Empirical evaluations on three benchmark data sets from different application domains demonstrated that BALPs generally outperform the state-of-the-art for plan recognition. We believe that the superior performance achieved by BALPs is due to the combination of logical abduction, joint probabilistic inference, and incorporation of domain knowledge. Overall, we found that BALPs were very effective for plan recognition.

4 Proposed Research

In this section, we discuss directions in which we propose to extend our work. For our next steps, we propose to apply BLPs to the task of machine reading. Our approach to machine reading involves learning general knowledge or commonsense rules about the domain. We then use these rules to infer additional information using the facts extracted by an off-the-shelf IE system. Our first aim involves learning the set of first-order rules using an existing ILP based rule learner and then applying BLPs for inference of additional facts. Our second aim involves developing a novel structure learner that learns a good set of first-order rules from uncertain, incomplete data. Our longer-term goals include extending our structure learner for multiple predicate learning, developing a novel method for discriminatively learning the parameters of BLPs and comparing the performance of BALPs to other probabilistic logics for plan recognition.

4.1 Extending BLPs for machine reading

The task of machine reading involves automatic extraction of knowledge from unstructured text. This information can later be used in other applications like question-answering. The internet has grown exponentially in the last few years, resulting in the accumulation of large amounts of on-line text. One way to search for information on a particular topic on the web is by using a search engine like Google[7] that returns relevant documents to the user. However, the user still has to read through all the documents to get the specific information he/she is looking for. Instead, it would be convenient to have a computer system that accepted a query/question from the user and returned the specific answer that the user is looking for. Such a system can be useful for several types of professionals like doctors who can use it to stay on top of the latest developments in medicine. People who work for security agencies can use it to keep track of various terroristic events that happen around the world. Due to these reasons, the last few years have attracted a lot of interest in machine reading.

One category of machine reading systems, also called information extraction (IE) systems (Cohen, 1999; Craven, DiPasquo, Freitag, McCallum, Mitchell, Nigam, & Slattery, 2000; Bunescu & Mooney, 2007; Etzioni, Banko, Soderland, & Weld, 2008) are trained to extract factual information that is stated explicitly in the text. Typically, in natural language text, several facts or commonsense information is not explicitly stated in the text. Since a human reader possesses the commonsense knowledge, he/she can understand from partial/incomplete information by inferring what is not present in the natural language text. But since IE systems do not possess this knowledge, they become incapable of extracting this implicit information. However, some queries require this type of implicit information to be returned as answers. Due to this reason, we find that traditional IE systems are limited in their ability to answer queries.

The other category of machine reading systems learn commonsense knowledge about the domain in the form of rules that capture the interactions and relations between different entities in the domain (Nahm & Mooney, 2000; Carlson, Betteridge, Kisiel, Settles, Jr., & Mitchell, 2010; Schoenmackers, Etzioni, Weld, & Davis, 2010; Doppa, NasrEsfahani, Sorower, Dietterich, Fern, & Tadepalli, 2010). These systems then use these rules to infer new information based on the facts stated explicitly in the text, thereby improving the performance of the underlying IE system. Nahm and Mooney (Nahm & Mooney, 2000) were the first group to learn propositional rules using C4.5 (Quinlan, 1996) from the output of an IE system on a computer-related job-postings domain. There are other systems (Carlson et al., 2010; Schoenmackers et al., 2010; Doppa et al., 2010) that learn first-order rules from natural language text. Carlson et al. (2010) and Doppa et al. (2010) modify existing ILP based rule learners like FOIL (Quinlan & Cameron-Jones, 1993) and FARMER (Nijssen & Kok, 2003) to learn probabilistic rules. These approaches use traditional logical deduction to infer additional facts. On the other hand, Schoenmackers et al. (2010) develop a new ILP system that uses statistical relevance to rank the learned rules. They use HOLMES (Schoenmackers, Etzioni, & Weld, 2008), an inference engine based on MLNs to infer additional facts.

We propose to develop a system for machine reading that can learn commonsense rules as described above. We use an off-the-shelf IE system to extract information from natural language text. We propose to apply BLPs for inference of additional facts. As described earlier, any probabilistic logic is better suited for the inference task on noisy data like natural language text than traditional logical deduction. Further, we demonstrated in Section 3 that due to the explosive grounding, MLNs do not always scale well to large

[7] `www.google.com`

domains. For these reasons, we hypothesize that our BLP approach for machine reading should be able to outperform the existing approaches.

4.1.1 Extending BLPs for Machine Reading

The standard inference mechanism in BLPs is SLD resolution. Given a knowledge base in the form of a set of first-order Horn clauses and a query, SLD resolution performs backchaining to generate deductive proofs that explain the given query. However, in the machine reading task we would like to infer additional facts given a set of extracted facts, which cannot be achieved using SLD resolution. As a result, we propose to modify the inference mechanism in BLPs by forward chaining on the extracted facts instead of performing backchaining.

4.1.2 Learning first-order rules for machine reading

We now discuss our approach to learning the structure of BLPs, i.e the set of first-order rules. The structure learner proposed by Kersting and DeRaedt (2008) assumes that every example in the training set is complete, i.e every example represents the least Herbrand model of the target BLP. However, this assumption does not hold for the extracted facts in machine reading for several reasons. First, not all facts are explicitly stated in the text. Second, there can be errors in the extraction process that can result in some facts not being extracted. As a result, the set of extracted facts are almost always incomplete, due to which we cannot use the structure learner developed by Kersting and DeRaedt for BLPs. There are several ILP based rules learners like FOIL (Quinlan & Cameron-Jones, 1993), PROGOL (Muggleton, 1995), ALEPH (Srinivasan, 2001), LIME (Mccreath & Sharma, 1998), etc. that can be used to learn the first-order rules. Most of these systems take a set of both positive and negative examples. Further, these rule learners cannot handle uncertainty in the examples. However, as explained earlier, uncertainty is inherently present in the examples extracted from natural language text and we do not have access to negative examples. For machine reading, we cannot apply the closed world assumption and assume any fact that is not extracted as a negative example since it might be implicit for the document. As a result, it is not possible to use most existing ILP based rule learners for our task. However, LIME is an existing ILP based rule learner that can induce rules only from positive examples. Further, it has some capability to handle noisy data, and hence we propose to use LIME to learn the first-order rules for the machine reading task. However, LIME or any existing rule learner does not have the capability to handle uncertain data, i.e it cannot take probability/confidence scores associated with extractions for learning rules.

We first identify a set of relations for which we would like to learn rules and then use LIME to learn rules for each target relation based on the remaining relations. We propose to learn rules for each relation independently, i.e the rules learned for one target relation are not used while learning rules for other target relations. We mentioned earlier that we decided to use LIME as it can learn only from positive instances and that we do not have access to negative instances for machine reading. However, LIME can also learn rules using both positive and negative instances. In order to learn all possible rules for the domain, we propose to learn rules in both the settings - 1) positive instances only 2) positive and negative instances. Typically, when negative instances are absent, closed world assumption is applied to automatically generate negative instances. According to the closed world assumption, any instance of a relation that is not an extracted fact can be considered a negative instance. While this is generally not true in machine reading, we still apply the closed world assumption to generate negative instances. As stated earlier, our goal is to learn all possible rules for the domain, and as a result, we do not want to limit to learning from positive instances only. We generate negative instances as follows: for each relation, we generate all possible instances using all constants in the domain without violating any type constraints. We then tag all instances that are not extracted facts (positive instances) as negative instances. If the set of negative instances generated is huge ($> 200,000$), then we sample instances from this set. We run LIME in the following modes - first, we give LIME only positive instances to learn rules. Then, we give LIME both positive and negative instances to learn rules. Since BLPs require the rules to be range restricted, we exclude all those rules that violate this property. We use the remaining set of rules for inference. Our proposed procedure for learning first-order rules is given in Algorithm 2

Algorithm 2 Learning first-order rules using LIME

```
Inputs: Set of target relations $R$ and the extracted facts for each example $E_j$ in the training set $T_r$
Output: Set of first-order rules $N$
1: $N$ is the set of first-order rules, initially empty
2: for each target relation $R_i$ in $R$ do
3:    $P \leftarrow$ set of all positive instances, i.e all instances of $R_i$ that are present in all $E_j$
4:    $N \leftarrow$ set of negative instances automatically generated
5:    $KB \leftarrow$ every extracted fact from every $E_j$ that is not an instance of $R_i$, i.e all the remaining extracted
      facts represent the background knowledge base
6:    $N_p \leftarrow$ run LIME with default settings to learn rules using only positive instances $P$. Pass $KB$ as the
      background knowledge base to LIME
7:    $N_n \leftarrow$ run LIME with default settings to learn rules using both positive instances $P$ and negative
      instances $N$. Pass $KB$ as the background knowledge base to LIME
8:    $N = N_p \cup N_n$
9:    for each rule $N_i$ in $N$ do
10:      if $N_i$ is not range restricted then
11:         Remove $N_i$ from $N$
12:      end if
13:   end for
14: end for
```

4.1.3 Learning parameters for BLPs

As described in Section 3.2, we propose to use logical-and and noisy-or models to specify the *cpd*s in the ground Bayesian network. We use a logical-and model to combine the evidence from the conjuncts in the body of clause. We use a noisy-or model to specify the *cpd* for combining the disjunctive contributions from different ground clauses with the same head. We propose to use the Expectation Maximization (EM) algorithm proposed by Kersting and DeRaedt (2008) to learn the noisy-or parameters for BLPs. If exact inference is intractable, then we will use SampleSearch (Gogate & Dechter, 2007), an approximate sampling algorithm for Bayesian networks with deterministic constraints (0 values in cpds). However, SampleSearch is a computationally expensive method that takes considerable time to generate sufficient number of samples. To speed up the learning process, we propose to use fewer examples for learning. Alternate approaches to parameter learning include the gradient ascent based approach for BLPs by Kersting and DeRaedt (2008) and hard EM. In hard EM, the values for the unobserved nodes are imputed using the MPE inference and the parameters are computed using frequency counting.

4.1.4 Evaluation

In this section, we describe our experimental setup to evaluate the BLP approach on machine reading. We propose to use SIRE, IBM's information extraction system (Florian, Hassan, Ittycheriah, Jing, Kambhatla, Luo, Nicolov, & Roukos, 2004) to extract facts from natural language text. We use DARPA's intelligent community (IC) data set for evaluation, which consists of news articles on various terroristic events around the world. There are 2000 documents in total, each containing an average of 89.9 extracted facts. DARPA has also specified the ontology for the IC domain that describes the entities and relations in the domain.

There are two approaches in which we propose to evaluate the quality of the learned rules. Since our primary goal is to improve the performance of the underlying IE system by inferring additional facts, we first evaluate the quality of the learned rules with respect to the quality of inferred facts. Second, we propose to evaluate the quality of the learned rules with respect to their ability to predict the target relation instances effectively. We now discuss the details of each evaluation strategy below.

- Evaluation with respect to the quality of the inferred facts
  In order to evaluate this aspect of our approach, we need to have access to all implicit and explicit facts that occur in a document. However, we do not have access to ground truth information. As a result, even if the BLP approach infers an additional fact, there is no automatic way of evaluating it. So, we propose to perform human evaluation for the inferred facts. One approach involves using Amazon's Mechanical Turk (Callison-Burch & Dredze, 2010) system to have the inferred facts evaluated by humans. The other approach involves sampling from the set of inferred facts and evaluating them ourselves. We compute the *estimated precision*, which measures the fraction of inferred facts that are correct.

- Evaluation with respect to the ability of the rules to predict the target relation instances
  For each target predicate, we eliminate all instances of it from the set of extracted facts in the test data set. We then use the learned rules to perform BLP inference over the set of remaining facts. Since the BLP inference is probabilistic in nature, every inferred fact is assigned a probability score that acts as a confidence score for the inferred fact. For various levels of confidence thresholds, ranging from 0.1 to 1, we compute the fraction of the eliminated instances that were inferred correctly by BLP, which we call the *estimated recall*. Note that we can never compute the *true* recall since we do not have ground truth information.

While the above approaches evaluate the quality of the rules induced and facts inferred, they do not necessarily evaluate if the BLP approach helps improve the performance of the underlying IE system with respect to answering queries automatically. We would like to evaluate the performance of the BLP approach for this application as well. For this evaluation, we propose to use the queries constructed for evaluation in the DARPA sponsored machine reading project (MRP). We hypothesize that the additional facts inferred using the BLP approach might result in answering more queries correctly than that answered by the underlying IE system.

Finally, we would like to compare the performance of BLPs to existing approaches like MLNs for machine reading. We can use the same set of first-order rules for both BLPs and MLNs. Alternatively, we can learn the structures for BLPs and MLNs using the respective structure learning methods. We learn the parameters for both BLPs and MLNs using their respective parameter learning approaches. We then perform inference using both probabilistic logics to infer additional facts. Finally, we evaluate the set of inferred facts using the approaches described above. We note that our proposed approach to evaluation is preliminary and we intend to extend it to additional domains in future.

politicalParty(A) ∧ employs(A,B) → hasMemberPerson(A,B) If a political party A employs B, then B is a member of A.
building(B) ∧ eventLocation(A,B) ∧ bombing(A) → thingPhysicallyDamaged(A,B) If a bombing event A took place in a building B, then B is physically damaged.
employs(A,B) → hasMember(A,B) If A employs B, then B is a member of A
citizenOf(A,B) → hasCitizenship(A,B) If A is a citizen of B, then A has the citizenship of B
nationState(B) ∧ person(A) ∧ employs(B,A) → hasBirthPlace(A,B) If a country B employs A, then A is born in B

Table 6: A sample set of rules learned using LIME

4.1.5 Preliminary results

We first identified a set of 18 relations including *employs(A,B)*, *eventLocation(Event, Loc)*, *hasGender(A,Gender)*, *hasMember(Group,Person)*, and *isLedBy(Org,Person)* for which first-order rules have to be learned. We split the documents into training set and test set in the ratio 9:1. We then used IBM's IE system to extract facts from the documents in both training and test set. As described in Procedure 2, we used LIME to learn first-order rules for target relations using the facts extracted from documents in the training set. We were able to learn rules for 13 out of 18 target relations. For the remaining 5 relations, even though we learned rules, the bodies of these rules consisted only entities. Such rules result in the inference of a lot of incorrect facts, and hence we systematically eliminated such rules from the learned structure. Figure 6 gives a set of interesting rules that were induced using LIME. In our preliminary analysis, we set the parameters for these rules manually. We used logical-and to combine the evidence from the conjuncts in the body of rule. We set all noisy-or parameters to a uniform values of 0.9 and we used maximum likelihood priors for all predicates. We then used the BLP approach to infer additional facts for the documents in the test set. In procedure 2, we used rules learned using only positive instances and both positive and negative instances for BLP inference. However, we also performed BLP inference using rules learned only from positive instances and those learned from both positive and negative instances. We call the model that uses both sets of rules BLP-All and the one that uses rules learned from only positive instances BLP-Pos-Only and the model that uses rules learned using both positive and negative instances BLP-Pos-Neg. The purpose of these experiments is to determine the impact of different rules on the quality of the inferred facts.

We used different sets of rules (BLP-All, BLP-Pos-Neg, and BLP-Pos-Only) to infer additional facts. In a purely logical approach, we forward chain on the rules based on the facts extracted from every document. However, in the BLP approach, we not only forward chain on the rules based on extracted facts, but also construct ground Bayesian networks for probabilistic inference. As a result, each fact inferred by the BLP approach is assigned a marginal probability, which is used as a threshold for evaluating inferred facts. In our approach, we used two thresholds - 0.9 and 0.95 for evaluation. Since we did not have access to the ground truth for the IC domain, we performed manual evaluation. For our preliminary analysis, we randomly sampled 20 documents from the test set and manually evaluated the inferred facts for the three models different and two thresholds.

Table 7 gives the preliminary results. We find that LIME is learning different types of rules when given only positive examples and when given both positive and negative examples. Even though the number of

Model	BLP-All	BLP-Pos-Neg	BLP-Pos-Only
No. of extracted facts	12254	12254	12254
No. of rules	40	27	14
Total no. of inferred facts (Purely logical approach)	38151	1510	18501
Estimated precision	24.94 (951/3813)*	16.00 (12/75)*	31.80 (588/1849)*
No. of inferred facts with prob $p >= 0.9$ (BLP-0.9)	12823	880	11717
Estimated precision	34.68 (419/1208)*	13.63 (9/66)*	35.49 (411/1158)*
No. of inferred facts with prob $p >= 0.95$ (BLP-0.95)	826	119	49
Estimated precision	91.07 (51/56)*	100 (1/1)*	- (0/0)*

Table 7: Estimated precision for facts inferred by BLP on the IC domain. * - x/y indicates that x out y inferred facts were correct. Note that estimated precision is calculated for facts inferred from 20 documents that were sampled randomly from the test set for manual evaluation.

rules induced by BLP-Pos-Neg model is higher than that induced by BLP-Pos-Only, the total number of facts inferred using the former is much lower than that inferred using the latter. This indicates that as expected, the rules learned from only positive instances are more general than those learned from both positive and negative instances. Across different models, we find that the number of facts inferred by the pure logical approach is higher than that inferred by the BLP approach for different thresholds. However, the estimated precision is very low. So, even though the purely logical approach improves recall, it does so at the cost of precision. On the other hand, the BLP approach gives the flexibility to use only those inferred facts with high confidence for subsequent analysis, thereby improving recall without sacrificing precision much. We find that the estimated precision for BLP-All to be reasonable at 0.95 confidence threshold. However, we find that only one fact was inferred by BLP-Pos-Neg and no facts inferred by BLP-Pos-Only at this confidence threshold. When we reduce the confidence threshold to 0.9, we find that the estimated precision for all models is very low.

Next, for each target relation, we eliminated all instances of it from the set of extracted facts in the test examples. We then ran the BLP inference over the remaining facts to infer additional facts. Finally, we evaluated if the eliminated instances were inferred by the BLP approach at various levels of confidence. Figure 3 gives the estimated recall averaged across all relations for different levels of confidence for BLP-All. We also show estimated recall for a few individual relations. We find that the average estimated recall for different confidence thresholds is fairly low. However, these results do not necessarily mean that the quality of the rules inferred is poor. Since the extracted facts are incomplete, it is possible that facts necessary to infer instances of target relations were missing, resulting in poor recall scores. Further, it might not be possible to infer some instances at all. For example, suppose we randomly eliminated a sentence from a document. Sometimes, due to redundancy in natural language text, it is possible to infer the sentence based on the other information present in the text. However, sometimes, it is not possible to infer this sentence at all. Similarly, in our context, some instances of the target relation can never be inferred, due to which we might see low estimated recall scores. We find that the accuracy for the target relation *AttendedSchool* is 100%. On closer inspection, there was only a single of instance of this relation in the test set, and it was inferred accurately, and hence the 100% accuracy.

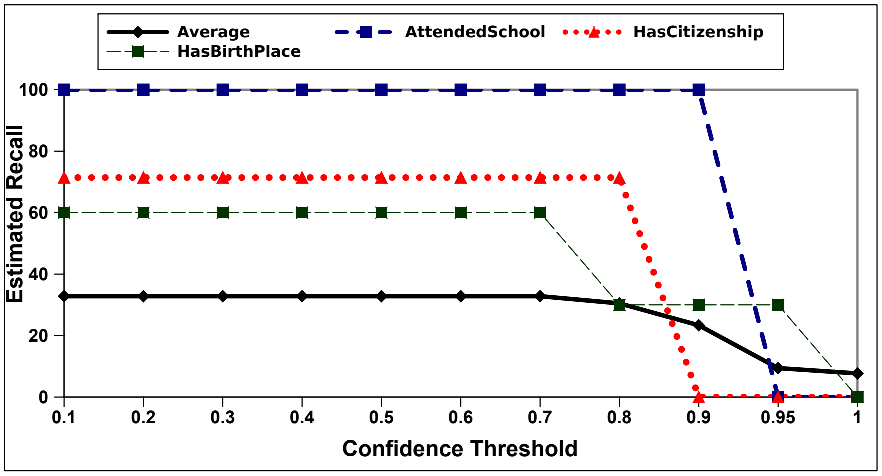

Figure 3: Estimated recall at different confidence thresholds for BLP-All.

4.2 Structure learning from uncertain and incomplete data

In this section, we discuss our proposed approach for learning the set of first-order rules for machine reading. In our approach, we propose to use an IE system to extract a set of facts from natural language text and then use these facts to induce first-order rules for the domain. As mentioned earlier, the extracted facts are incomplete and noisy. Instead of using this noisy data, we can use the ground truth data for learning the rules. However, we do not have access to the ground truth data for the IC domain and manual annotation is both expensive and time consuming. Further, since the input to BLP inference is the set of noisy extractions from the IE system, it would be better to induce rules using these noisy extractions, rather than using the ground truth. For every extracted fact, the IE system generates a probability score that reflects the confidence of the IE system in that extraction, which we propose to use for ranking learned rules.

We will now discuss relevant work on the topic of learning first-order rules. We have already discussed different ILP based rule learners in Section 4.1.2. These methods suffer from the following limitations. First, most of them require a complete set of positive and negative instances to learn rules. Second, all of them including LIME (Mccreath & Sharma, 1998) have limited capability to deal with noisy data, but none of them can handle uncertainty in the data, i.e they cannot accept confidence scores for examples. As a result, none of the existing ILP based rule learners are suited for learning first-order rules for machine reading. There are several structure learning algorithms developed for directed probabilistic logics like BLPs (Kersting & Raedt, 2008), Logical Bayesian Networks (LBNs) (Ramon, Croonenborghs, Fierens, & Blockeel, 2007), Probabilistic Relational Models (PRMs) (Getoor, Friedman, Koller, & Pfeffer, 2001), and First-Order Conditional Influence (FOCI) (Natarajan, keen Wong, & Tadepalli, 2006). However, most of these algorithms assume that the examples are complete and have no capability to deal with uncertain data. As mentioned earlier, there are other systems (Carlson et al., 2010; Schoenmackers et al., 2010; Doppa et al., 2010) that learn first-order rules for the task of machine reading. None of these algorithms use the confidence scores of extractions for learning rules.

We propose to develop a method that overcomes the limitations of existing methods for learning first-order rules or structure of BLPs. First, the structure learner should be able to induce rules only from positive examples, since we do not have access to negative examples, and since the closed world assumption does not hold for this domain. Second, since the data is incomplete and uncertain, the structure learner should be able to deal with it. One way to deal with uncertainty is by incorporating the confidence scores or probabilities for extractions while inducing rules. To the best of our knowledge, no rule leaner or structure learner that has the capability to incorporate uncertainty in examples has been developed. We propose to employ bottom-up approaches (Mihalkova & Mooney, 2007; Kok & Domingos, 2009) like relational path finding (Richards & Mooney, 1992) as these approaches have demonstrated superior performance over the traditional top-down approaches (Kok & Domingos, 2005; Kersting & Raedt, 2008; Quinlan & Cameron-Jones, 1993) for learning rules. Most structure learners learn rules for a single target predicate at a time. While this approach is sufficient for learning rules for the task of classification, it is not ideal for the machine reading task. However, learning rules for multiple predicates is a very difficult task (De Raedt et al., 1993), more so for our domain as it involves incomplete and uncertain data. As a result, we will simplify the problem by learning first-order rules for a single target predicate/relation in the immediate future.

4.3 Longer-term future work

We now discuss directions in which we propose to extend our work in the longer term.

4.3.1 Multiple Predicate Learning

Existing methods for structure learning almost always learn first-order rules for a single target relation at a time since learning multiple predicates is a hard problem (De Raedt et al., 1993). Even in our proposed approach for learning first-order rules, we attempt to learn rules for a single predicate at a time. We believe that we would be able to learn a better structure if we learned rules for all target predicates at the same time. In the longer–term, we propose to extend our approach to multiple predicate learning.

4.3.2 Discriminative parameter learning for BLPs

Kersting and De Raedt (2008) propose two different algorithms for learning the parameters of a BALP from data – Expectation Maximization and gradient ascent based learning. Both methods learn parameters by optimizing the data log-likelihood, which make them unsuitable for tasks that involve classification. We propose to explore approaches that either optimize the conditional likelihood (Greiner et al., 2002) or perform max-margin based learning (Huynh & Mooney, 2009) of parameters for BLPs so that the learned model would be better suited for classification tasks. To the best of our knowledge, no algorithm that learns the parameters discriminatively for BLPs has been developed till date.

4.3.3 Comparison of BALPs to other probabilistic logics for plan recognition

In this proposal, we compare the performance of BALPs to that of MLNs on the plan recognition task. There are other probabilistic logics like Abductive Stochastic Logic Programs (ASLPs) (Tamaddoni-Nezhad et al., 2006), PRISM (Sato, 1995), Poole's Horn Abduction (Poole, 1993) that perform abductive reasoning. However, these methods have not been applied to abductive plan recognition. We propose to compare the performance of BALPs with these systems in the future.

5 Conclusions

Research in the area of statistical relational learning (SRL) has gained a lot of attention in the last few years due to the advantages of SRL formalisms for handling noisy structured data. Bayesian Logic Programs

(BLPs), which combine first-order logic and Bayesian networks are a simple, yet powerful formalism for solving problems involving structured data. In this proposal, we apply BLPS to two real world tasks – plan recognition and machine reading. First, we extended BLPs to the task of abductive plan recognition and we call the resulting model Bayesian Abductive Logic Programs (BALPs). Experimental evaluation on benchmark data sets showed that BALPs outperformed the state-of-the-art for plan recognition.

For future work, we apply BLPs to the task machine reading. Our approach to machine reading involves learning general knowledge or commonsense rules about the domain. We then use these rules to infer additional information using the facts extracted by an off-the-shelf IE system. Our first aim involves learning the set of first-order rules using an existing ILP based rule learner and then applying BLPs for the inference of additional facts. Our second aim involves developing a novel structure learner that learns a good set of first-order rules from uncertain and incomplete data. Our longer-term goals include extending our structure learner for multiple predicate learning, developing a novel method for discriminative learning of parameters for BLPs and comparing the performance of BALPs to other probabilistic logics on plan recognition.

6 Acknowledgement

We thank Parag Singla and Pradeep Ravikumar for their valuable inputs for implementing the EM algorithm for parameter learning. We also thank Vibhav Gogate for helping modify SampleSearch for our experiments. This research was partly funded by MURI ARO grant W911NF-08-1-0242. This research was also funded by Air Force Contract FA8750-09-C-0172 under the DARPA Machine Reading Program. Experiments were run on the Mastodon Cluster, provided by NSF grant EIA-0303609.

References

Albrecht, D. W., Zukerman, I., & Nicholson, A. E. (1998). Bayesian models for keyhole plan recognition in an adventure game. In *User Modeling and User-Adapted Interaction*, pp. 5–47.

Blaylock, N., & Allen, J. (2005a). Generating artificial corpora for plan recognition. In *International Conference on User Modeling (UM-05)*. Springer.

Blaylock, N., & Allen, J. (2005b). Recognizing instantiated goals using statistical methods. In *G. Kaminka (Ed.), Workshop on Modeling Others from Observations (MOO-05)*, pp. 79–86.

Blaylock, N., & Allen, J. F. (2004). Statistical goal parameter recognition. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS-04)*, pp. 297–305.

Bui, H. H. (2003). A general model for online probabilistic plan recognition. In *Proceedings of the Eighteenth International Joint Conference on Artificial intelligence (IJCAI-03)*.

Bunescu, R. C., & Mooney, R. J. (2007). Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07)*, Prague, Czech Republic.

Callison-Burch, C., & Dredze, M. (2010). Creating speech and language data with Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pp. 1–12, Stroudsburg, PA, USA. Association for Computational Linguistics.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Jr., E. H., & Mitchell, T. (2010). Toward an architecture for never-ending language learning. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pp. 1306–1313. AAAI Press.

Charniak, E., & Goldman, R. (1991). A probabilistic model of plan recognition. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pp. 160–165, Anaheim, CA.

Charniak, E., & Goldman, R. P. (1989). A semantics for probabilistic quantifier-free first-order languages, with particular application to story understanding. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, Detroit, MI.

Charniak, E., & McDermott, D. (1985). *Introduction to Artificial Intelligence*. Addison-Wesley, Reading, MA.

Chen, J., Muggleton, S., & Santos, J. (2008). Learning probabilistic logic models from probabilistic examples. *Machine Learning*, *73*(1), 55–85.

Cohen, W. W. (1999). Whirl: A word-based information representation language. *Artificial Intelligence*, *118*, 163–196.

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., & Slattery, S. (2000). Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence*, *118*(1-2), 69–113.

De Raedt, L., & Dehaspe, L. (1997). Clausal discovery. *Machine Learning*, *26*(2–3), 99–146.

De Raedt, L., & Kersting, K. (2004). Probabilistic inductive logic programming. In *Algorithmic Learning Theory*, pp. 19–36.

De Raedt, L., Lavrac, N., & Dzeroski, S. (1993). Multiple predicate learning. In *IJCAI*, pp. 1037–1043.

Doppa, J. R., NasrEsfahani, M., Sorower, M. S., Dietterich, T. G., Fern, X., & Tadepalli, P. (2010). Towards learning rules from natural texts. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading (FAM-LbR 2010)*, pp. 70–77, Stroudsburg, PA, USA. Association for Computational Linguistics.

Elvira-Consortium (2002). Elvira: An environment for probabilistic graphical models. In *Proceedings of the Workshop on Probabilistic Graphical Models*, Cuenca, Spain.

Etzioni, O., Banko, M., Soderland, S., & Weld, D. S. (2008). Open information extraction from the web. *Communications of ACM*, *51*, 68–74.

Florian, R., Hassan, H., Ittycheriah, A., Jing, H., Kambhatla, N., Luo, X., Nicolov, N., & Roukos, S. (2004). A statistical model for multilingual entity detection and tracking. In *HLT-NAACL*, pp. 1–8.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, Stockholm, Sweden.

Getoor, L., & Taskar, B. (Eds.). (2007). *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA.

Getoor, L., Friedman, N., Koller, D., & Pfeffer, A. (2001). Learning probabilistic relational models. In Džeroski, S., & Lavrač, N. (Eds.), *Relational Data Mining*.

Gogate, V., & Dechter, R. (2007). Samplesearch: A scheme that searches for consistent samples. In *Proceedings of Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS-07).*

Greiner, R., Su, X., Shen, B., & Zhou, W. (2002). Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. In *Proceedings of the Eighteenth Annual National Conference on Artificial Intelligence (AAAI-02)*, pp. 167–173.

Heckerman, D., & Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. In *Machine Learning*, pp. 20–197.

Horvitz, E., & Paek, T. (1999). A computational architecture for conversation. In *Proceedings of the Seventh International Conference on User Modeling*, pp. 201–210. Springer.

Huber, M. J., Durfee, E. H., & Wellman, M. P. (1994). The automated mapping of plans for plan recognition. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pp. 344–351. Morgan Kaufmann.

Huynh, T. N., & Mooney, R. J. (2009). Max-margin weight learning for Markov logic networks. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-09)*, pp. 564–579, Bled, Slovenia.

Huynh, T. N., & Mooney, R. J. (2008). Discriminative structure and parameter learning for Markov logic networks.. pp. 416–423, Helsinki, Finland.

Kakas, A. C., Kowalski, R. A., & Toni, F. (1993). Abductive logic programming. *Journal of Logic and Computation*, *2*(6), 719–770.

Kate, R. J., & Mooney, R. J. (2009). Probabilistic abduction using Markov logic networks. In *Proceedings of the IJCAI-09 Workshop on Plan, Activity, and Intent Recognition*, Pasadena, CA.

Kautz, H. A. (1987). *A Formal Theory of Plan Recognition*. Ph.D. thesis, Department of Computer Science, University of Rochester, Rochester, NY. Technical Report 215.

Kautz, H. A., & Allen, J. F. (1986). Generalized plan recognition. In *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pp. 32–37, Philadelphia, PA.

Kersting, K., & De Raedt, L. (2007). Bayesian logic programming: Theory and tool. In Getoor, L., & Taskar, B. (Eds.), *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA.

Kersting, K., & De Raedt, L. (2001). Towards combining inductive logic programming with Bayesian networks. In *Proceedings of the 11th International Conference on Inductive Logic Programming (ILP-2001)*, pp. 118–131, Strasbourg, France.

Kersting, K., & Raedt, L. D. (2008). Basic principles of learning bayesian logic programs. In *Probabilistic Inductive Logic Programming*, pp. 189–221.

Koivisto, M., & Sood, K. (2004). Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, *5*, 549–573.

Kok, S., & Domingos, P. (2005). Learning the structure of Markov logic networks. In *Proceedings of 22nd International Conference on Machine Learning (ICML-2005)*, Bonn,Germany.

Kok, S., & Domingos, P. (2009). Learning Markov logic network structure via hypergraph lifting.. pp. 505–512, Montreal, Quebec, Canada.

Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Lauritzen, S. L. (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, *19*, 191–201.

Lesh, N., & Etzioni, O. (1995). A sound and fast goal recognizer. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*.

Levesque, H. J. (1989). A knowledge-level account of abduction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pp. 1061–1067, Detroit, MI.

Mccreath, E., & Sharma, A. (1998). Lime: A system for learning relations. In *Ninth International Workshop on Algorithmic Learning Theory*, pp. 336–374. Springer-Verlag.

Mihalkova, L., & Mooney, R. J. (2007). Bottom-up learning of Markov logic network structure. In *Proceedings of 24th International Conference on Machine Learning (ICML-2007)*, Corvallis, OR.

Muggleton, S. (2000). Learning stochastic logic programs. In *Proceedings of the AAAI2000 Workshop on Learning Statistical Models from Relational Data*.

Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing*, *13*, 245–286.

Muggleton, S. (2003). Learning structure and parameters of stochastic logic programs. In *Proceedings of the twelfth international conference on Inductive logic programming (ILP-02)*, pp. 198–206, Berlin, Heidelberg. Springer-Verlag.

Muggleton, S. H. (Ed.). (1992). *Inductive Logic Programming*. Academic Press, New York, NY.

Nahm, U. Y., & Mooney, R. J. (2000). A mutually beneficial integration of data mining and information extraction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00)*, pp. 627–632, Austin, TX.

Natarajan, S., keen Wong, W., & Tadepalli, P. (2006). Structure refinement in first order conditional influence language. In *Proceedings of the Workshop on Open Problems in Statistical Relational Learning (SRL)*.

Nau, D., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., & Yaman, F. (2003). Shop2: An HTN planning system. *Journal of Artificial Intelligence Research*, *20*, 379–404.

Ng, H. T., & Mooney, R. J. (1990). The role of coherence in abductive explanation. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pp. 337–442, Detroit, MI.

Ng, H. T., & Mooney, R. J. (1992). Abductive plan recognition and diagnosis: A comprehensive empirical evaluation. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pp. 499–508, Cambridge, MA.

Nijssen, S., & Kok, J. N. (2003). Efficient frequent query discovery in farmer. In *Proceedings of the seventh conference in Principles and Practices of Knowledge Discovery in Database (PKDD 2003)*, pp. 350–362. Springer.

Nilsson, D. (1998). An efficient algorithm for finding the M most probable configurations in probabilistic expert systems. *Statistics and Computing*, *8*, 159–173.

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

Peirce, C. S. (1958). *Collected Papers of Charles Sanders Peirce*. MIT Press, Cambridge, Mass.

Perlich, C., & Merugu, S. (2005). Multi-relational learning for genetic data: Issues and challenges. In Džeroski, S., & Blockeel, H. (Eds.), *Fourth International Workshop on Multi-Relational Data Mining (MRDM-2005)*.

Poole, D. (1993). Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, *64*, 81–129.

Pople, H. E. (1973). On the mechanization of abductive logic. In *Proceedings of the Third International Joint Conference on Artificial Intelligence (IJCAI-73)*, pp. 147–152.

Pynadath, D. V., & Wellman, M. P. (2000). Probabilistic state-dependent grammars for plan recognition. In *In Proceedings of the Conference on Uncertainty in Artificial Intelligence, UAI2000*, pp. 507–514. Morgan Kaufmann Publishers.

Quinlan, J. R. (1996). Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp. 725–730, Portland, OR.

Quinlan, J. R., & Cameron-Jones, R. M. (1993). FOIL: A midterm report. In *Proceedings of the European Conference on Machine Learning*, pp. 3–20, Vienna.

Raghavan, S., & Mooney, R. (2010). Bayesian abductive logic programs. In *Proceedings of the AAAI-10 Workshop on Statistical Relational AI (Star-AI 10)*, pp. 82–87, Atlanta, GA.

Ramon, J., Croonenborghs, T., Fierens, D., & Blockeel, H. (2007). Generalized ordering-search for learning directed probabilistic logical models. In *Machine Learning*.

Richards, B. L., & Mooney, R. J. (1992). Learning relations by pathfinding. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pp. 50–55, San Jose, CA.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*, 107–136.

Rosner, B. (2005). *Fundamentals of Biostatistics*. Duxbury Press.

Russell, S., Binder, J., Koller, D., & Kanazawa, K. (1995). Local learning in probabilistic networks with hidden variables. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 1146–1152, Montreal, Canada.

Russell, S., & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach* (2 edition). Prentice Hall, Upper Saddle River, NJ.

Saria, S., & Mahadevan, S. (2004). Probabilistic plan recognition in multiagent systems. In *International Conference on Automated Planning and Scheduling (ICAPS 2004)*.

Sato, T. (1995). A statistical learning method for logic programs with distribution semantics. In *Proceedings of the twelfth international conference on logic programming (ICLP-95)*, pp. 715–729. MIT Press.

Schoenmackers, S., Etzioni, O., & Weld, D. S. (2008). Scaling textual inference to the web. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pp. 79–88, Stroudsburg, PA, USA. Association for Computational Linguistics.

Schoenmackers, S., Etzioni, O., Weld, D. S., & Davis, J. (2010). Learning first-order Horn clauses from web text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pp. 1088–1098, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shapiro, E. Y. (1983). *Algorithmic Program Debugging*. MIT Press, Cambridge, MA.

Silander, T., & Myllymäki, P. (2006). A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of Uncertainty in Artificial Intelligence*.

Singla, P., & Mooney, R. (2011). Abductive Markov logic for plan recognition. In *Twenty-fifth National Conference on Artificial Intelligence (to appear)*.

Srinivasan, A. (2001). *The Aleph manual*. `http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/`.

Stickel, M. E. (1988). A Prolog-like inference system for computing minimum-cost abductive explanations in natural-language interpretation. Tech. rep. Technical Note 451, SRI International, Menlo Park, CA.

Tamaddoni-Nezhad, A., Chaleil, R., Kakas, A., & Muggleton, S. (2006). Application of abductive ilp to learning metabolic network inhibition from temporal data. *Machine Learning*, *64*(1-3), 209–230.

Teyssier, M., & Koller, D. (2005). Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proceedings of Uncertainty in Artificial Intelligence*, pp. 584–590.