Plan Recognition using Statistical Relational Models

Sindhu Raghavan[a], Parag Singla[b], Raymond J. Mooney[a]

[a]*Department of Computer Science*
The University of Texas at Austin
2317 Speedway, Stop D9500
Austin, TX 78712, USA

[b]*Department of Computer Science and Engineering*
Indian Institute of Technology Delhi
Hauz Khas, New Delhi 110016, INDIA

Abstract

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 plans that best *explain* observed actions. Most existing approaches to plan recognition and other abductive reasoning tasks either use first-order logic (or subsets of it) or probabilistic graphical models. While the former cannot handle uncertainty in the data, the latter cannot handle structured representations. To overcome these limitations, we explore the application of statistical relational models that combine the strengths of both first-order logic and probabilistic graphical models to plan recognition. Specifically, we introduce two new approaches to abductive plan recognition using Bayesian Logic Programs (BLPs) and Markov Logic Networks (MLNs). Neither of these formalisms is suited for abductive reasoning because of the deductive nature of the underlying logical inference. In this work, we propose approaches to adapt both these formalisms for abductive plan recognition. We present an extensive evaluation of our approaches on three benchmark datasets on plan recognition, comparing them with existing state-of-the-art methods.

Email addresses: sindhu.vijayaraghavan@gmail.com (Sindhu Raghavan), parags@cse.iitd.ac.in (Parag Singla), mooney@cs.utexas.edu (Raymond J. Mooney)

Keywords: Plan Recognition, Abduction, Reasoning about Actions, Probabilistic Reasoning, Statistical Relational Learning, Bayesian Logic Programs, Markov Logic Networks, Machine Learning

1. Introduction

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 and McDermott, 1985). Early approaches to plan recognition were based on first-order logic in which a knowledge-base of plans and actions is developed for the domain and then default reasoning (Kautz and Allen, 1986; Kautz, 1987) or logical abduction (Ng and Mooney, 1992) is used to predict the best plan based on the observed actions. Kautz and Allen (1986); Kautz (1987) developed one of the first logical formalizations of plan recognition. They used non-monotonic deductive inference to predict plans using observed actions, an action taxonomy, and a set of commonsense rules or constraints. Lesh and Etzioni (1995)'s approach 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. The approach by Hong (2001) also constructs a "goal graph" and analyses the graph to identify goals consistent with 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.

Another approach to plan recognition is to directly use probabilistic methods. Albrecht et al. (1998) developed an approach based on dynamic Bayesian networks to predict plans in an adventure game. Horvitz and Paek (1999) developed an approach that uses Bayesian networks to recognize goals in an automated conversation system. Pynadath and Wellman (2000) extended probabilistic context-free grammars to plan recognition. Kaminka et al. (2002) developed an approach to multiagent plan recognition using dynamic Bayesian networks to perform monitoring in distributed systems. Bui et al. (2002); Bui (2003) used Abstract Hidden Markov Models for hierarchical goal recognition. Saria and Mahadevan (2004) extended the work by Bui (2003) to multiagent plan recognition. Blaylock and Allen (2005) used statistical n-gram models for the task of instantiated goal recognition. While these approaches can handle uncertainty and can be trained effec-

tively, they can not handle the kind of structured relational data that can be represented in first-order predicate logic. Furthermore, it is difficult to incorporate planning domain knowledge in these approaches.

The third category of approaches use aspects of both logical as well as probabilistic reasoning. Hobbs et al. (1988) attach weights or costs to predicates in the knowledge base and use these weights to guide the search for the best explanation. Goldman et al. (1999) use the probabilistic Horn abduction (Poole, 1993) framework to find the best set of plans that explain the observed actions. Several other approaches use Bayesian networks (Charniak and Goldman, 1989, 1991; Huber et al., 1994) to perform abductive inference. Based on the observed actions and a knowledge base constructed for planning, Bayesian networks are automatically constructed using knowledge base model construction (KBMC) procedures. However, most of these approaches do not have the capabilities for learning the structure or the parameters. Another chapter by Inoue et al. (2013) appearing in the current volume explores the use of weighted abduction for large scale discourse processing.

The last decade has seen a rapid growth in the area of *Statistical Relational Learning* (SRL) (Getoor and Taskar, 2007), which uses well-founded probabilistic methods while maintaining the representational power of first-order logic. Since these models combine the strengths of both first-order logic and probabilistic graphical models, we believe that they are well suited for solving problems like plan recognition. In this paper, we explore the efficacy of different SRL models for the task of plan recognition. We focus on extending two specific SRL models, Markov Logic Networks (MLNs) (Domingos and Lowd, 2009) (based on undirected probabilistic graphical models) and Bayesian Logic Programs (BLPs) (Kersting and De Raedt, 2001) (based on directed probabilistic graphical models) to the task of plan recognition.

MLNs attach real-valued weights to formulas in first-order logic in order to represent their certainty. They effectively use logic as a compact template for representing large, complex ground Markov networks. Since MLNs have been shown to formally subsume many other SRL models and have been successfully applied to many problems (Domingos and Lowd, 2009), we chose to explore their application to plan recognition. However, the representational power and flexibility offered by MLNs come at a cost in computational complexity. In particular, many problems result in exponentially large ground Markov networks, making learning

and inference intractable in the general case.

Pearl (1988) argued that causal relationships and abductive reasoning from effect to cause are best captured using directed graphical models (Bayesian networks). Since plan recognition is abductive in nature, this suggested that we also explore a formalism based on directed models. Therefore, we also explored the application of BLPs, which combine first-order Horn logic and directed graphical models, to plan recognition. BLPs use SLD resolution to generate proof trees, which are then used to construct a ground Bayesian network for a given query. This approach to network construction is called knowledge base model construction (KBMC). Similar to BLPs, prior approaches (Wellman et al., 1992; Ngo and Haddawy, 1997) also employ the KBMC technique to construct ground Bayesian networks for inference. Another approach called probabilistic Horn abduction (PHA) (Poole, 1993) performs adductive reasoning using first-order knowledge bases and Bayesian networks. However, since the BLP framework imposes fewer constraints on representation, both with respect to structure as well as the parameters (Kersting and De Raedt, 2007) and since it provides an integrated framework for both learning and inference, we decided to use BLPs as opposed to PHA or other similar formalisms.

Logical approaches to plan recognition, e.g. (Kautz and Allen, 1986; Ng and Mooney, 1992), typically assume a knowledge base of plans and/or actions appropriate for planning, but not specifically designed for plan recognition. The advantage of this approach is that a single knowledge base is sufficient for both automated planning and plan recognition. Also, knowledge of plans and actions sufficient for planning is usually easier to develop than a knowledge base especially designed for plan recognition, which requires specific rules of the form "If an agent performs action A, they may be executing plan P." A recent MLN-based approach to plan/activity recognition (Sadilek and Kautz, 2010b,a), requires such manually-provided plan-recognition rules.

Our goal is to develop general-purpose SRL-based plan-recognition systems that only require the developer to provide a knowledge-base of actions and plans sufficient for planning, without the need to engineer a knowledge-base specifically designed for plan recognition. Plan recognition using only planning knowledge generally requires *abductive* logical inference. BLPs use purely deductive logical inference as a pre-processing step to the full blow-blown probabilistic inference. Encoding the planning knowledge directly in MLNs does not support abductive

reasoning either. Further, using the standard semantics of MLNs of grounding the whole theory leads to a blow-up for plan recognition problems. Consequently, neither BLPs or MLNs can be directly used in their current form for abductive plan recognition. Therefore, this paper describes re-encoding strategies (for MLNs) as well as enhancements to both models that allow them to utilize logical abduction. Our other goal involves developing systems that are capable of learning the necessary parameters automatically from data. Since both BLPs and MLNs provide algorithms for learning both the structure and the parameters, we adapt them in our work to develop trainable systems for plan recognition.

The main contributions of the paper are as follows:

- Adapt SRL models like BLPs and MLNs to plan recognition
- Introduce Bayesian Abductive Logic Programs (BALPs), an adaptation of BLPs that utilizes logical abduction
- Propose re-encoding strategies for facilitating abductive reasoning in MLNs
- Introduce abductive Markov logic, an adaptation of MLNs which combines re-encoding strategies with logical abduction to construct the ground Markov network
- Experimentally evaluate the relative performance of BALPs, abductive MLNs (i.e. using re-encoding strategies and abductive model construction), traditional MLNs, and existing plan-recognition methods on three plan-recognition benchmarks.

The rest of the paper is organized as follows. First, we provide some background on logical abduction, BLPs, and MLNs. Next, we present our extensions to both BLPs and MLNs to include logical abduction. Finally, we present an extensive evaluation of our approaches on three benchmark datasets for plan recognition, comparing them with the existing state-of-the-art for plan recognition.

2. Background

2.1. Logical Abduction

In a logical framework, abduction is usually defined as follows (Pople, 1973; Levesque, 1989; Kakas et al., 1993):

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

Here, $\models$ represents logical entailment and $\bot$ represents 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 given set of observations O. Following Occam's Razor, the best hypothesis is typically defined as the one that minimizes the number of assumptions, $|H|$. Given a set of observations O_1, O_2, ..., O_n, the set of abductive proof trees is computed by recursively backchaining on each O_i until every literal in the proof is either proven or assumed. Logical abduction has been applied to tasks such as plan recognition and diagnosis (Ng and Mooney, 1992; Peng and Reggia, 1990).

2.2. Bayesian Logic Programs

Bayesian logic programs (BLPs) (Kersting and De Raedt, 2001) can be viewed as templates for constructing *directed* graphical models (Bayes nets). Given a knowledge base as a special kind of logic program, standard backward-chaining logical deduction (SLD resolution) is used to automatically construct a Bayes net on the same lines as knowledge based model construction (KBMC) (Wellman et al., 1992; Breese et al., 1994). 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 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 (Pearl, 1988). 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: 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 (Koller and Friedman, 2009). Typically, we would like to compute the most probable explanation (MPE), which finds the joint assignment of values to unobserved nodes in the network that maximizes the posterior probability given the values of a set of observed nodes. This type of inference is also called also known as the maximum a posteriori (MAP) assignment and might be used interchangeably in this book. We would also like to compute the marginal probabilities for the unobserved nodes given the values of observed nodes. The combining-rule parameters and cpd entries for a BLP can be learned automatically from data using techniques proposed by Kersting and De Raedt (2008).

2.3. Markov Logic Networks

Markov logic (Richardson and Domingos, 2006; Domingos and Lowd, 2009) is a framework for combining first-order logic and undirected probabilistic graphical models (Markov networks). A traditional first-order knowledge base can be seen as a set of hard constraints on the set of possible worlds: if a world violates even one formula, its probability is zero. In order to soften these constraints, Markov logic attaches a weight to each formula in the knowledge base. A formula's weight reflects how strong a constraint it imposes on the set of possible worlds. Formally, an MLN is a set of pairs (F_i, w_i), where F_i is a first-order formula and w_i is a real number. A *hard clause* has an infinite weight and acts as a logical constraint; otherwise, it is a *soft clause*. Given a set of constants, an MLN defines a ground Markov network with a node in the network for each ground atom and a feature for each ground clause. The joint probability distribution over a set of boolean variables $X = (X_1, X_2...)$ corresponding to the nodes in the ground Markov network (i.e. ground atoms) is defined as:

$$P(X = x) = \frac{1}{Z} exp(\sum_i w_i n_i(x)) \tag{1}$$

where $n_i(x)$ is the number of true groundings of F_i in x and Z is a normalization term obtained by summing $P(X = x)$ over all values of X. Therefore, a possible world becomes exponentially less likely as the total weight of the ground clauses it violates increases.

An MLN can be viewed as a set of templates for constructing ground Markov networks. Different sets of constants produce different Markov networks; however, there are certain regularities in their structure and parameters determined by the underlying first-order theory. Like in BLPs, once the ground network is constructed, standard probabilistic inference methods (MPE or marginal inference) can be used to answer queries. MLN weights can be learned by maximizing the conditional log-likelihood of training data supplied in the form of a database of true ground literals. A number of efficient inference and learning algorithms that exploit the structure of the network have also been proposed. Domingos and Lowd (2009) provide details on these and many other aspects of MLNs.

3. Adapting Bayesian Logic Programs

Bayesian Abductive Logic Programs (BALPs) are an adaptation of BLPs. In plan recognition, the known facts are insufficient to support the derivation of deductive proof trees for the requisite queries. By using *ab*duction, missing literals can be assumed in order to complete the proof trees needed to determine the structure of the ground network. We first describe the abductive inference procedure used in BALPs. Next we describe how probabilistic parameters are specified and how probabilistic inference is performed. Finally, we discuss how parameters can be automatically learned from data.

3.1. Logical Abduction

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 for each such rule. 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, Abduction-BALP 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 minimize the number

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 use 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) Partial knowledge base with two rules for road block:

1. $\texttt{blk\_rd}(\texttt{Loc}) | \texttt{hvy\_snow}(\texttt{Loc}), \texttt{drive\_hzrd}(\texttt{Loc})$.
2. $\texttt{blk\_rd}(\texttt{Loc}) | \texttt{acdnt}(\texttt{Loc}), \texttt{clr\_wrck}(\texttt{Crew}, \texttt{Loc})$.

(b) Observations:

$\texttt{blk\_rd}(\texttt{plaza})$

(c) Ground Abductive Clauses:

$\texttt{blk\_rd}(\texttt{plaza}) | \texttt{hvy\_snow}(\texttt{plaza}), \texttt{drive\_hzrd}(\texttt{plaza})$.
$\texttt{blk\_rd}(\texttt{plaza}) | \texttt{acdnt}(\texttt{plaza}), \texttt{clr\_wrck}(\texttt{a1}, \texttt{plaza})$.

Figure 1: (a) A partial knowledge base from the emergency response domain in the Monroe data set. All variables start with uppercase and constants with lowercase. (b) The logical representation of the observations. (c) The set of ground rules obtained from logical abduction.

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, i.e. when multiple assumptions match with a subgoal, we randomly pick one of them and do not pursue the others. We found that this approach worked well for our plan-recognition benchmarks. 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 motivated by one of our evaluation benchmarks, the emergency response domain introduced by Blaylock and Allen (2005) in the Monroe data set described in Section 5.1. Consider the partial knowledge base and set of observations given in Fig-

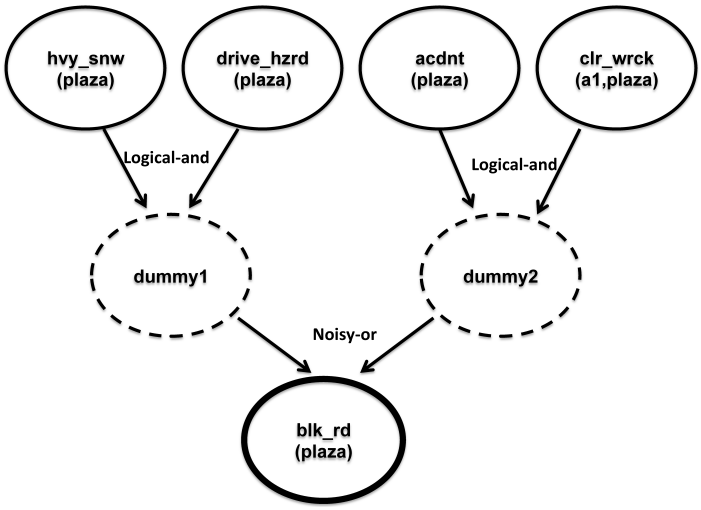

Figure 2: Bayesian network constructed for example in Figure 1. The nodes with thick borders represent observed actions, the nodes with dotted borders represent intermediate nodes used to combine the conjuncts in the body of a clause, and the nodes with thin borders represent plan literals.

ure 1a and Figure 1b respectively. The knowledge base consists of rules that give two explanations for a road being blocked at a location: 1) there has been heavy snow resulting in hazardous driving conditions, and 2) there has been an accident and the crew is clearing the wreck. Given the observation that a road is blocked, the plan recognition task involves abductively inferring one of these causes as the explanation. For each observation literal in Figure 1b, we recursively backchain to generate abductive proof trees. In the given example, we observe that the road is blocked at the location $plaza$. When we backchain on the literal $\texttt{blk\_rd}(\texttt{plaza})$ using Rule 1, we obtain the subgoals $\texttt{hvy\_snow}(\texttt{plaza})$ and $\texttt{drive\_hzrd}(\texttt{plaza})$. These subgoals become assumptions since no observations or heads of clauses unify with them. We then backchain on the literal $\texttt{blk\_rd}(\texttt{plaza})$ using Rule 2 to

obtain subgoals $\texttt{acdnt}(\texttt{plaza})$ and $\texttt{clr\_wrk}(\texttt{Crew}, \texttt{plaza})$. Here again, we find that these subgoals have to be assumed since there are no facts or heads of clauses that unify with them. We further notice that $\texttt{clr\_wrk}(\texttt{Crew}, \texttt{plaza})$ is not a fully ground instance. Since $\texttt{Crew}$ is an existentially quantified variable, we replace it with a Skolem constant $a1$ to get the ground assumption $\texttt{clr\_wrk}(\texttt{a1}, \texttt{plaza})$.

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 ($\texttt{hvy\_snow}(\texttt{plaza})$, $\texttt{drive\_hzrd}(\texttt{plaza})$, $\texttt{acdnt}(\texttt{plaza})$, and $\texttt{clr\_wrk}(\texttt{Crew}, \texttt{plaza})$) 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.

3.2. Probabilistic Modeling, Inference, and Learning

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. Further, techniques developed for BLPs for learning parameters can also be used for BALPs.

We now discuss how parameters are specified in BALPs. We use noisy/logical-and and noisy-or models to specify the cpds 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 logical-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 the most probable explanation (MPE) inference (Pearl, 1988). We compute multiple alternative explanations using the k-MPE algorithm (Nilsson, 1998) as implemented in the ELVIRA Elvira-Consortium (2002) package.

Learning can be used to automatically set the noisy-or and noisy-and parameters

in the model. 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 use the EM algorithm adapted for BLPs by Kersting and De Raedt (2008). We simplify the problem by learning only noisy-or parameters and using a deterministic logical-and model to combine evidence from the conjuncts in the body of a clause.

4. Adapting Markov Logic

As previously mentioned, encoding the planning knowledge directly in MLNs does not support abductive reasoning. This is because of the deductive nature of the rules encoding the planning knowledge. In MLNs, the probability of a possible world increases with the total weight of the satisfied formulae. Since an implication is satisfied whenever its consequent is true, an MLN is unable to abductively infer the antecedent of a rule from its consequent. Given the rule $P \Rightarrow Q$ and the observation that Q is true, we would like to abduce P as a possible cause for Q. Since the consequent (Q) is true, the clause is trivially satisfied, independent of the value of the antecedent (P), and hence does not give any information about the truth value of the antecedent (P).

In this section, we describe three key ideas for adapting MLNs with logical abductive reasoning, each one building on the previous. First, we describe the Pairwise Constraint (PC) model proposed by Kate and Mooney (2009). Next, we introduce the Hidden Cause (HC) model, which alleviates some of the inefficiencies of the PC model. These two models offer strategies for the re-encoding MLN rules but do not change the semantics of the traditional MLNs (Richardson and Domingos, 2006). Next, we introduce an abductive model construction procedure on top of the HC model that results in even simpler Markov networks. This gives us the formulation for abductive Markov logic. Our ground Markov network construction strategy is different from the one used in traditional MLNs, and hence, our formulation results in a different semantics.

We also describe an alternate approach to plan recognition in which the structure

of the MLN is manually encoded to enable deductive inference of the top-level plans from observed actions. This allows us to compare abductive Markov logic to a manually encoded MLN for plan recognition.

4.1. Pairwise Constraint Model

Kate and Mooney (2009) were the first to develop an approach to re-encode MLNs with logical abductive reasoning, which we call the Pairwise Constraint (PC) model. We describe this approach here since it provides the context for understanding the more sophisticated models introduced in subsequent sections. The key idea is to introduce explicit reversals of the implications appearing in the original knowledge base. Multiple possible explanations for the same observation are supported by having a disjunction of the potential explanations in the reverse implication. "Explaining away" (Pearl, 1988) (inferring one cause eliminates the need for others) is achieved by introducing a mutual-exclusivity constraint between every pair of possible causes for an observation. Given the set of Horn clauses: $P_1 \Rightarrow Q, P_2 \Rightarrow Q, \cdots P_n \Rightarrow Q$, a reverse implication: $Q \Rightarrow P_1 \vee P_2 \cdots \vee P_n$, and a set of mutual-exclusivity constraints: $Q \Rightarrow \neg P_1 \vee \neg P_2, \cdots Q \Rightarrow \neg P_{n-1} \vee \neg P_n$ for all pairs of explanations are introduced. The weights on these clauses control the strength of the abductive inference and the typical number of alternate explanations, respectively. We do not need to explicitly model these constraints in BLPs, since the underlying model is Bayesian networks which capture the full conditional probability distribution (CPD) of each node given its parents and the mutual exclusivity constraints are implicitly modeled in the conditional distribution. For first-order Horn clauses, all variables not appearing in the head of the clause become existentially quantified in the reverse implication. We refer the reader to Kate and Mooney (2009) for the details of the conversion process.

We now illustrate the PC model with the same example described in Section 3. It is an example from one of our evaluation benchmarks, the emergency response domain introduced by Blaylock and Allen (2005) (all variables start with uppercase and constants with lowercase, and by default variables are universally quantified):

$\texttt{hvy\_snow}(\texttt{Loc}) \wedge \texttt{drive\_hzrd}(\texttt{Loc}) \Rightarrow \texttt{blk\_rd}(\texttt{Loc})$
$\texttt{acdnt}(\texttt{Loc}) \wedge \texttt{clr\_wrk}(\texttt{Crew}, \texttt{Loc}) \Rightarrow \texttt{blk\_rd}(\texttt{Loc})$

These rules give two explanations for a road being blocked at a location: 1) there has been heavy snow resulting in hazardous driving conditions, and 2) there has been an accident and the crew is clearing the wreck. Given the observation that a road is blocked, the plan recognition task involves abductively inferring one of these causes as the explanation. Using the PC model, we get the final combined reverse implication and pairwise constraint clauses as below:

$$\texttt{blk\_rd}(\texttt{Loc}) \Rightarrow (\texttt{hvy\_snow}(\texttt{Loc}) \wedge \texttt{drive\_hzrd}(\texttt{Loc})) \vee (\exists \texttt{Crew acdnt}(\texttt{Loc}) \wedge \texttt{clr\_wrk}(\texttt{Crew}, \texttt{Loc}))$$

$$\texttt{blk\_rd}(\texttt{Loc}) \Rightarrow \neg(\texttt{hvy\_snow}(\texttt{Loc}) \wedge \texttt{drive\_hzrd}(\texttt{Loc})) \vee \neg(\exists \texttt{Crew acdnt}(\texttt{Loc}) \wedge \texttt{clr\_wrk}(\texttt{Crew}, \texttt{Loc}))$$

The first rule introduces the two possible explanations and the second rule constrains them to be mutually exclusive.

The PC model can construct very complex networks since it includes multiple clause bodies in the reverse implication, making it very long. If there are n possible causes for an observation and each of the corresponding Horn clause has k literals in its body, then the reverse implication has $O(nk)$ literals. This in turn results in cliques of size $O(nk)$ in the ground network. This significantly increases the computational complexity since probabilistic inference is *exponential* in the treewidth of the graph, which in turn is at least the size of the maximum clique (Koller and Friedman, 2009). The PC model also introduces $O(n^2)$ pairwise constraints, which can result in a large number of ground clauses. As a result, the PC model does not generally scale well to large domains.

4.2. Hidden Cause Model

The Hidden Cause (HC) model alleviates some of the inefficiencies of the PC model by introducing a hidden cause node for each possible explanation. The joint constraints can then be expressed in terms of these hidden causes, thereby reducing the size of the reverse implication (and hence, the corresponding clique size) to $O(n)$. The need for the pairwise constraints is eliminated by specifying a low prior on all hidden causes. A low prior on the hidden causes indicates that the hidden causes are most likely to be false, unless there is explicit evidence indicating the their presence. Hence, given an observation, inferring one cause obviates the need for the others. We now describe the HC model more formally. We first consider the propositional case for the ease of explanation. It is extended

to first-order Horn clauses in a straightforward manner. Consider the following set of rules describing the possible explanations for a predicate Q:

$P_{i1} \wedge P_{i2} \wedge \cdots \wedge P_{ik_i} \Rightarrow Q, \forall i, (1 \leq i \leq n)$

For each rule we introduce a hidden cause C_i and add the following rules to the MLN:

1. $P_{i1} \wedge P_{i2} \wedge \cdots \wedge P_{ik_i} \Leftrightarrow C_i, \forall i, (1 \leq i \leq n)$ (soft)
2. $C_i \Rightarrow Q, \forall i, (1 \leq i \leq n)$ (hard)
3. $Q \Rightarrow C_1 \vee C_2 \cdots C_n$ (reverse implication) (hard)
4. $\texttt{true} \Rightarrow C_i, \forall i, (1 \leq i \leq n)$ (negatively weighted) (soft)

The first set of rules are soft clauses with high positive weights. This allows the antecedents to sometimes fail to cause the consequent (and vice-versa). The next two sets of rules are hard clauses, they implement a deterministic-or function between the consequent and the hidden causes. The last one is a soft rule and implements a low prior (by having a negative MLN weight) on the hidden causes. These low priors discourage inferring multiple hidden causes for the same consequent ("explaining way"), and the strength of the prior determines the degree to which multiple explanations are allowed.

Different sets of weights on the biconditional in the first set of rules implement different ways of combining multiple explanations. For example, a noisy-or model (Pearl, 1988) can be implemented by modeling the implication from antecedents to the hidden cause as a soft constraint and the reverse direction as a hard constraint. The weight w_i for the soft-constraint is set to $log[(1 - p_{f_i})/p_{f_i}]$, where p_{f_i} is the failure probability for cause i. This formulation is related to previous work on combining functions in Markov logic (Natarajan et al., 2010); however, we focus on the use of such combining functions in the context of abductive reasoning. There has also been prior work on discovering hidden structure in a domain (e.g. (Davis et al., 2007; Kok and Domingos, 2007)). However, in our case, since we know the structure of the hidden predicates in advance, there is no need to discover them using the methods referenced above.

We now describe how to extend this approach to first-order Horn clauses. For first-order Horn clauses, variables present in the antecedents but not in the consequent become existentially quantified in the reverse implication, as in the PC model. But

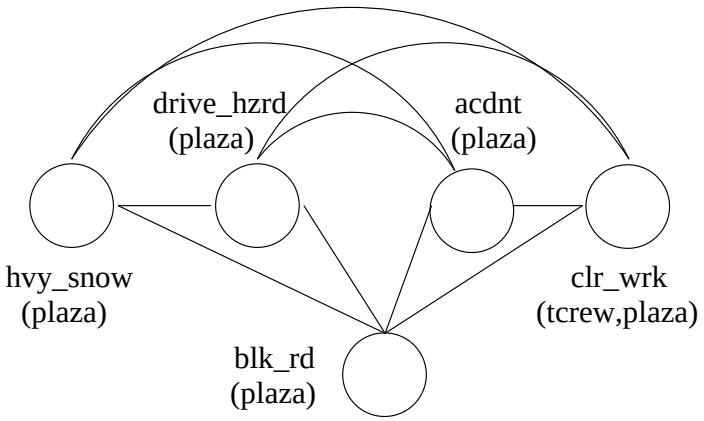

Figure 3: Ground network constructed by the PC model for the road blocked example

unlike the PC model, the reverse implication expression is much simpler as it only involves one predicate (the hidden cause) for each rule implying the consequent. Let us revisit the example from the emergency response domain. Based on the HC approach, we introduce two hidden causes (rb_C1(Loc) and rb_C2(Loc)) corresponding to the two (hard) rules:

$$\texttt{hvy\_snow}(\texttt{Loc}) \wedge \texttt{drive\_hzrd}(\texttt{Loc}) \Leftrightarrow \texttt{rb\_C1}(\texttt{Loc})$$
$$\texttt{acdnt}(\texttt{Loc}) \wedge \texttt{clr\_wrk}(\texttt{Crew}, \texttt{Loc}) \Leftrightarrow \texttt{rb\_C2}(\texttt{Crew}, \texttt{Loc})$$

Note that each hidden cause contains all variables present in the antecedent of the rule. These hidden causes are combined with the original consequent using the following (soft) rules:

$$\texttt{rb\_C1}(\texttt{Loc}) \Rightarrow \texttt{blk\_rd}(\texttt{Loc})$$
$$\texttt{rb\_C2}(\texttt{Crew}, \texttt{Loc}) \Rightarrow \texttt{blk\_rd}(\texttt{Loc})$$
$$\texttt{blk\_rd}(\texttt{Loc}) \Rightarrow \texttt{rb\_C1}(\texttt{Loc}) \vee \exists \texttt{Crew}(\texttt{rb\_C2}(\texttt{Crew}, \texttt{Loc}))$$

In addition, there are (soft) unit clauses specifying low priors on the hidden causes.

Figures 3 and 4 show the ground networks constructed by the PC and HC models respectively, when `loc` is bound to `Plaza` and `crew` to `Tcrew`. The PC model results in a fully connected graph (maximum clique size is 5), whereas the HC model is much sparser (maximum clique size is 3). Consequently, inference in the HC model is significantly more efficient.

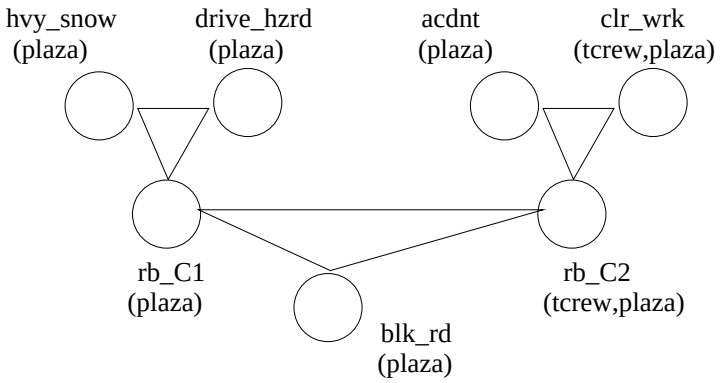

Figure 4: Ground network constructed by the HC model for the road blocked example

Algorithm 2 presents the pseudocode for constructing the re-encoded MLN using the HC approach for a given Horn-clause knowledge base. In lines 2 to 8, hidden causes are created for each possible explanation for each consequent (line 5). A biconditional is introduced between the hidden causes and the corresponding antecedents (line 6), which is modeled as a soft clause in the MLN. Each hidden cause is also linked to the corresponding consequent via a hard clause (line 7). The next part (lines 9 to 24) combines the hidden causes for each of the consequents into a single reverse implication. The rules are partitioned according to the first-order predicate appearing in the consequent (line 9). For each partition (line 10), each possible instantiation of the consequent predicate appearing in the underlying rules is considered (lines 11 to 13). For instance, given the rule: $\mathtt{h}_1(\mathtt{X},\mathtt{Y}) \Rightarrow \mathtt{q}(\mathtt{Y})$ and another: $\mathtt{h}_2(\mathtt{X},\mathtt{const}) \Rightarrow \mathtt{q}(\mathtt{const})$, we need to consider each instantiation of $\mathtt{q}(\mathtt{X})$ and $\mathtt{q}(\mathtt{const})$ separately. For each such instantiation c (line 14), we consider the rules which could result in the consequent c being true (line 15). Technically, these are rules whose consequents subsume c, i.e. there exists a substitution θ_i such that $c = C(r_i)\theta_i$, where $C(r_i)$ is the consequent of rule r_i. These rules result in c when bound by the substitution θ_i. For each such rule r_i (line 17), substitution θ_i is applied to the corresponding hidden cause $H(r_i)$ (line 18). Then, the set of free variables in the hidden cause (i.e. the variables not appearing in c) is extracted (line 19). These variables are existentially quantified in the reverse implication (next step). We then introduce a reverse implication to indicate that c implies at least one of the consequents (amongst those that subsume c) (line 21). This reverse implication is made a hard clause and added to the MLN (line 22). Finally, a low prior is introduced for each hidden cause (lines 25 to 27).

Algorithm 2 GenReEncodedMLN(**KB**)

inputs: **KB**, background knowledge consisting of Horn clauses
output: $\mathbf{M}$, set of rules in the re-encoded MLN

$\mathbf{M} \leftarrow \{\}$
for all $r \in$ **KB do**
  $A(r) \leftarrow$ antecedent in r
  $C(r) \leftarrow$ consequent in r
  $H(r) \leftarrow$ hidden cause for r
  $\mathbf{M} \leftarrow \mathbf{M} \cup \{A(r) \Leftrightarrow H(r)\}$
  $\mathbf{M} \leftarrow \mathbf{M} \cup \{H(r) \Rightarrow C(r)\}$
end for
Part(**KB**) $\leftarrow$ partition of **KB** into sets or rules with same (first-order) predicate in the consequent
for all set of rules $R \in$ Part(**KB**) **do**
  Let $R = \{r_1, r_2, \cdots r_m\}$
  $C(R) \leftarrow \bigcup_{i=1}^{m} \{C(r_i)\}$
  ($C(R)$ is set of unique consequents appearing in R)
  for all $c \in C(R)$ **do**
    $R_c \leftarrow \{r_i \in R \mid \exists \theta_i, c = C(r_i)\theta_i\}$
    (R_c is set of rules whose consequents subsume c)
    for all $r_i \in R_c$ **do**
      $H_{\theta_i}(r_i) \leftarrow H(r_i)\theta_i$
      $\{v_{i_1}, v_{i_2}, \cdots v_{i_k}\} \leftarrow$ variables appearing in $H_{\theta_i}(r_i)$ but not in c
    end for
    $I_r(R_c) \leftarrow (c \Rightarrow \bigvee_{i=1}^{m} \exists v_{i_1}, v_{i_2}, \cdots v_{i_k} H_{\theta_i}(r_i))$
    $\mathbf{M} \leftarrow \mathbf{M} \cup \{I_r(R_c)\}$
  end for
end for
for all $r \in$ **KB do**
  $\mathbf{M} \leftarrow \mathbf{M} \cup \{\texttt{true} \Rightarrow H(r)$ (negatively weighted) $\}$
end for
return M

4.3. Abductive Model Construction

Abductive reasoning using MLNs consists of the following 3 steps: 1) Generate the re-encoded MLN, 2) Construct the ground Markov network (model construc-

tion), 3) Perform learning/inference over the resulting ground network. The standard MLN model construction process uses the set of all possible ground atoms (the Herbrand base) and constructs the set of all possible ground clauses using these atoms. Using logical rules to construct a graphical model is generally referred to as *knowledge-based model construction* (KBMC), originally proposed by Wellman (Wellman et al., 1992; Breese et al., 1994). This idea was further used by Ngo and Haddawy (1997) for answering queries from probabilistic knowledge bases. In abductive reasoning, we are looking for explanations for a given set of observations. In this task, the set of all possible ground atoms and ground clauses may not be needed to explain the observations. Considering the fully ground network leads to increased time and memory complexity for learning and inference. For instance, in the road blocked example, if the observation of interest is $\texttt{blk\_rd}(\texttt{plaza})$, then we can ignore groundings where the location is not `plaza`.

We propose an alternative model-construction procedure that uses logical abduction to determine the set of *relevant* ground atoms. The procedure first constructs the set of abductive proof trees for the observations and then uses only the ground atoms in these proofs instead of the full Herbrand base for constructing the ground network. The ground Markov network is then constructed by instantiating the formulae in the abductive MLN using this reduced set of ground atoms. We refer to the set of ground atoms (Markov network) thus constructed as the abductive ground atoms (Markov network). First, given a set of Horn rules and a set of observations, the rules for the abductive MLN are constructed using the HC model. Next, the set of most-specific abductive proof trees for the observations are computed using the method of Stickel (1988). The abductive inference process to get the set of most-specific abductive proofs is described in Algorithm 1. The atoms in these proofs form the set of abductive ground atoms. For each formula in the abductive MLN, the set of all ground formulae whose atoms appear in the abductive ground atoms are added to the ground Markov network. While handling existentials, only those disjuncts which belong to the set of abductive ground atoms are used. Learning and inference are then performed over the resulting network.

In general, the abductive model construction procedure results in a ground network that is substantially different (and usually much simpler) from that constructed using the full Herbrand base. It also differs from the network constructed by starting KBMC from the query/observations (Domingos and Lowd, 2009) because of the use of the backward chaining and unification during the abductive model construction. Consequently, the probabilistic inferences supported by

this model can be different from that of the traditional MLN model. This also makes the abductive process different from other pre-processing approaches such as Shavlik and Natarajan (2009), or existing lifted inference techniques such as Singla and Domingos (2008), both of which produce a network that is probabilistically equivalent to the original. By focusing on the relevant ground atoms, abductive model construction significantly improves the performance of abductive MLNs *both* in terms of time and memory efficiency as well as predictive accuracy. Further, lifted inference could still be applied by constructing a lifted network over the nodes/clauses present in the abductive network. For consistency with Singla and Mooney (2011), we will refer to abductive MLNs (using the Hidden Cause model followed by abductive model construction) as MLN-HCAM.

Detailed experiments demonstrating significantly improved plan-recognition performance (with respect to both accuracy and efficiency) for the HC model and abductive model construction are presented by Singla and Mooney (2011). Therefore, we only compare with the MLN-HCAM approach in the experiments below.

4.4. Plan Recognition using Manually Encoded MLNs

As discussed in the introduction, traditional (non-abductive) MLNs can be used for plan recognition if an appropriate set of clauses are manually provided that directly infer higher-level plans from primitive actions, e.g. (Sadilek and Kautz, 2010b,a). In order to compare to this approach, we developed an MLN approach that uses a manually-engineered knowledge base to perform deductive plan recognition.

The clauses for this manually encoded MLN were constructed as follows. For each top-level plan predicate, we identified any set of observations that uniquely identifies this plan. This implies that no other plan explains this set of observations. We then introduced a rule that infers this plan given these observations and we make it a hard clause in the MLN. If no such observations exist, we introduced a soft rule for each observation that could potentially indicate this plan. Hard mutual-exclusivity rules were included for plans when we know only one of them can be true. Alternatively, we gave a negative prior to all plan predicates as described in the MLN-HCAM model. Note that this approach does not include any predicates corresponding to sub-goals that are never observed in the data. As a result, all variables in this model are fully observed during training, resulting in a

less complex model for learning and inference. This way of encoding the domain knowledge avoids the automated machinery for introducing reverse implications, and can potentially lead to a simpler knowledge base, which results in simpler ground networks. However, it requires a separate knowledge-engineering process that develops explicit plan-recognition rules for the domain, while the abductive approach only requires basic knowledge about plans and actions sufficient for planning. There are several ways in which such explicit plan-recognition knowledge bases can be manually engineered, and we have employed just one possible approach. Exploring alternative approaches is a direction for future work. Subsequently, we refer to this manually encoded traditional MLN model as "MLN-manual".

4.5. Probabilistic Modeling, Inference, and Learning

In MLN-HC and MLN-HCAM, we use the noisy-or model to combine multiple explanations. In all MLN models, we use the cutting-plane method (Riedel, 2008) for MPE inference and MC-SAT (Poon and Domingos, 2006) to compute marginal probabilities of plan instances. For learning weights of the clauses in MLN-HCAM, we use a version of gradient-based voted-perceptron algorithm (Singla and Domingos, 2005) modified for partially observed data as discussed in Chapter 20 (Section 3.3.1) of Koller and Friedman (2009). For the traditional MLN model, gradient-based voted-perceptron algorithm runs out of memory due to the size and complexity of the resulting ground networks. Therefore, we learned weights using the discriminative online learner proposed by Huynh and Mooney (2011a).[1] More details about various settings used in our experiments are described in Section 5.

5. Experimental Evaluation

In this section, we present an extensive experimental evaluation of the performance of BALPs and MLNs on three plan-recognition data sets. Unfortunately,

[1] This is possible since the training data for this model is fully observed (i.e there are no hidden nodes).

there have been very few rigorous empirical evaluations of plan-recognition systems, and there are no widely-used benchmarks. Our experiments employ three extant data sets and compare to the state-of-the-art results in order to demonstrate the advantages of SRL methods. In addition to presenting concrete results for BALPs and MLNs, we also consider their relationship to other probabilistic logics, discussing the relative strengths and weaknesses of different SRL models for plan recognition.

5.1. Datasets

5.1.1. Story Understanding

We first used a small dataset previously used to evaluate abductive story understanding systems (Ng and Mooney, 1992; Charniak and Goldman, 1991).[2] 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 (in English) 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. Most 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 planning background knowledge initially constructed for the ACCEL system (Ng and Mooney, 1992), which contains a total of 126 Horn rules.

5.1.2. Monroe

We also used the Monroe dataset, an artificially generated plan-recognition dataset in the emergency response domain by Blaylock and Allen (2005). This domain includes 10 top level plans like setting up a temporary shelter, clearing a road wreck, and providing medical attention to victims. The task is to infer a single instantiated top-level plan based on a set of observed actions automatically generated by a *hierarchical transition network* (HTN) planner. We used the the logical plan

[2] Available at `http://www.cs.utexas.edu/users/ml/accel.html`

knowledge base for this domain constructed by Raghavan and Mooney (2011), which consists of 153 Horn clauses. We used 1,000 artificially generated examples in our experiments. Each example instantiates one of the 10 top-level plans and contains an average of 10.19 ground literals describing a sample execution of this plan.

5.1.3. 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 plan schemas and 457 examples, with an average of 6.1 command literals per example. Here again, we used the plan knowledge base constructed by Raghavan and Mooney (2011), which consists of 50 Horn clauses.

Each of these data sets evaluates a distinct aspect of plan recognition systems. Since the Monroe domain is quite large with numerous subgoals and entities, it tests the ability of a plan-recognition system to scale to large domains. On the other hand, the Linux data set is not that large, but since the data comes from real human users, it is quite noisy. There are several sources of noise including cases in which users claim that they have successfully executed a top-level plan when actually they have not (Blaylock and Allen, 2005). Therefore, this data set tests the robustness of a plan-recognition system to noisy input. Monroe and Linux involve predicting a *single* top-level plan; however, in the Story Understanding domain, most examples have multiple top-level plans. Therefore, this data set tests the ability of a plan-recognition system to identify multiple top-level plans.

5.2. Comparison of BALPs, MLNs, and Existing Approaches

In this section, we present experimental results comparing the performance of BALPs and MLNs to that of existing plan recognition approaches like ACCEL (Ng and Mooney, 1992) and the system developed by Blaylock and Allen (2005). ACCEL is a purely logical abductive reasoning system that uses a variable evaluation metric to guide its search for the best explanation. It can use two differ-

ent evaluation metrics: *simplicity*, which selects the explanation with the fewest assumptions, and *coherence*, which selects the explanation that maximally connects the input observations (Charniak and Goldman, 1990). This second metric is specifically geared towards text interpretation by measuring *explanatory coherence* (Ng and Mooney, 1990). Currently, this bias has not been incorporated in either BALPs or any of the MLN approaches. Blaylock and Allen's system is a purely probabilistic system that learns statistical n-gram models to separately predict plan schemas (i.e. predicates) and their arguments.

Section 4 describes several variants of MLNs for plan recognition. All MLN models were implemented using Alchemy (Kok et al., 2010), an open source software package for learning and inference in Markov logic. We used the logical abduction component developed for BALPs (Algorithm 1) for abductive model construction in MLNs. Since this abductive Markov-logic formulation (MLN-HCAM) performs better than simple re-encodings of the traditional MLN, we restrict our comparative experiments to this approach. For more details on the experimental results comparing the different MLN enhancements, we refer the reader to Singla and Mooney (2011). For the Monroe and Linux datasets, we also compare with the traditional (non-abductive) MLN approach described in Section 4.4, referred to as "MLN-manual."

For BALPs, we learned the noisy-or parameters using the EM algorithm described in Section 3.2 whenever possible. Similarly for both MLN-HCAM and MLN, we learned the parameters using the algorithms described in Section 4.5. For datasets that had multiple top plans as the explanation, we computed the most probable explanation (MPE). For datasets that had a single correct top-level plan, we computed the marginal probability of plan instances and picked the one with the highest marginal probability. For both MLNs and BALPs, we have used exact inference whenever feasible. However, when exact inference was intractable, we used approximate sampling to perform probabilistic inference – Sample Search (Gogate and Dechter, 2011) for BALPs and MC-SAT (Poon and Domingos, 2006) for MLNs. Both these techniques are approximate sampling algorithms designed for graphical models with deterministic constraints. Whenever we deviate from this standard methodology, we provide the specific details.

5.2.1. Story Understanding

This section provides information on the methodology and results for experiments on the story-understanding data set.

Parameter Learning
For BALPs, we were unable to learn effective parameters from the mere 25 development examples. As a result, we set parameters manually in an attempt to maximize performance on the development set. A uniform value of 0.9 for all noisy-or parameters seemed to work well for this domain. The intuition behind our choice of noisy-or parameters is as follows: if a parent node is true, then with a probability of 0.9, the child node will be true. In other words, if a cause is true, then the effect is true with the probability of 0.9. Using the deterministic logical-and model to combine evidence from conjuncts in the body of a clause did not yield high-quality results. Using noisy-and significantly improved the results; so we used noisy-and's with uniform parameters of 0.9. Here again, the intuition is that if parent node is false or turned off, then the child node would also be false or turned off with a probability 0.9. To disambiguate between conflicting plans, we set different priors for high level plans to maximize performance on the development data. For MLN-HCAM, we were able to learn more effective weights from the examples in the development set using the learning algorithm described in the Section 4.5.

Probabilistic Inference
Since multiple plans are possible in this domain, we computed the most probable explanation (MPE) to infer the best set of plans. Since the resulting graphical models for this domain were not exceptionally large, we were able to apply exact (rather than approximate) inference algorithms. For BALPs, we used the k-MPE algorithm (Nilsson, 1998) as implemented in Elvira (Elvira-Consortium, 2002). For MLN-HCAM, we used with the cutting-plane method (and the associated code) developed by Riedel (2008).

Evaluation Metrics
We compared BALPs and MLN-HCAM with ACCEL-Simplicity and ACCEL-Coherence. We compared the inferred plans with the ground truth to compute *precision* (the percentage of the inferred plans that are correct), *recall* (the percentage of the correct plans that were properly inferred), and *F-measure* (the har-

	Precision	Recall	F-measure
ACCEL-Simplicity	66.45	52.32	58.54
ACCEL-Coherence	89.39	89.39	89.39
BALP	72.07	85.57	78.24
MLN-HCAM	69.13	75.32	72.10

Table 1: Results for the Story Understanding dataset

monic mean of precision and recall). Partial credit was given for predicting the correct plan predicate with some incorrect 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 score is 66.67%.

Results
Table 1 shows the results for Story Understanding. Both BALPs and MLN-HCAM perform better than ACCEL-Simplicity, demonstrating the advantage of SRL models over standard logical abduction. BALPs perform better than MLN-HCAM, demonstrating an advantage of a directed model for this task. However, ACCEL-Coherence still gives the best results. 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. Incorporating this bias into SRL models is difficult since it uses the graphical structure of the abductive proof to evaluate an explanation, which is not straightforward to include in a probabilistic model. It should be noted that the coherence metric is specific to narrative interpretation, since it assumes the observations are connected together into a coherent "story," and this assumption is not generally applicable to other plan recognition problems.

5.2.2. Monroe and Linux

This section provides information on the methodology and results for experiments on the Monroe and Linux data sets.

Parameter Learning
For BALPs, we learned the noisy-or parameters automatically from data using the EM algorithm described in Section 3.2. We initially set all noisy-or parameters

to 0.9, which gave reasonable performance in both domains. We picked a default value of 0.9 based on the intuition that if a parent node is true, then the child node is true with a probability 0.9. We then ran EM with two starting points – random weights and manual weights (0.9). We found that EM initialized with manual weights generally performed the best for both domains, and hence we used this approach for our comparisons. Even though EM is sensitive to its starting point, it outperformed other approaches even when initialized with random weights. For Monroe and Linux, initial experiments found no advantage to using noisy-and instead of logical-and in these domains, so we did not experiment with learning noisy-and parameters.

For MLN-HCAM, we learned the weights automatically from data using the methods described in Section 4.5. For MLN-manual, the online weight learner did not provide any improvement over default manually-encoded weights (a weight of 1 for all the soft clauses and a weight of -0.5 on unit clauses for all plan predicates to specify a small prior for all plans). Therefore, we report results obtained using these manually-encoded weights.

For Monroe, of the 1,000 examples in our dataset we used the first 300 for training, the next 200 for validation, and the remaining 500 examples for test. Blaylock and Allen learned their parameters on 4,500 artificially generated examples. However, we found that using a large number of examples resulted in much longer training times and that 300 examples were sufficient to learn effective parameters for both BALPs and MLN-HCAM. For BALPs, we ran EM on the training set until we saw no further improvement in the performance on the validation set. For MLN-HCAM, the parameter learner was limited to training on at most 100 examples, as learning on larger amounts of data ran out of memory. Therefore, we trained MLN-HCAM on 3 disjoint subsets of the training data and picked the best model using the validation set.

For Linux, we performed 10-fold cross validation. For BALPs, we ran EM until convergence on the training set for each fold. For MLN-HCAM, within each training fold, we learned the parameters on disjoint subsets of data, each consisting of around 110 examples. As mentioned before for Monroe, the parameter learner did not scale to larger data sets. We then used the rest of the examples in each training fold for validation, picking the model that best performed on the validation set.

As discussed in section 4.4, in the traditional MLN model, there are two ways to

encode the bias that only a single plan is needed to explain a given action. The first approach is to include explicit hard mutual-exclusivity constraints between competing plans, the second approach involves setting a low prior on all plan predicates. While the former performed better on Monroe, the latter gave better results on Linux. We report the results of the best approach for each domain.

As noted, we had to adopt a slightly different training methodology for BALPs and MLN-HCAM due to computational limitations of MLN weight learning on large datasets. However, we used the exact same test sets for both systems on all datasets. Developing more scalable online methods for training MLNs on partially observable data, is an important direction for future work.

Probabilistic Inference
Both Monroe and Linux involve inferring a single top-level plan. Therefore, we computed the marginal probability of each plan instantiation and picked the most probable one. For BALPs, since exact inference was tractable on Linux, we used the exact inference algorithm implemented in Netica,[3] a commercial Bayes-net software package. For Monroe, the complexity of the ground network made exact inference intractable and we used SampleSearch (Gogate and Dechter, 2011), an approximate sampling algorithm for graphical models with deterministic constraints. For both MLN approaches, we used MC-SAT (Poon and Domingos, 2006) as implemented in the Alchemy system on both Monroe and Linux.

Evaluation Metric
We compared the performance of BALPs, MLN-HCAM and MLN with Blaylock and Allen's system using the *convergence score* as defined by Blaylock and Allen (2005). The convergence score measures the fraction of examples for which the correct plan predicate is inferred (ignoring the arguments) when given *all* of the observations. Use of the convergence score allowed for the fairest comparison to the original results on these datasets published by Blaylock and Allen.[4]

Results
Table 2 shows convergence score for both the Monroe and Linux datasets. Both

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

[4] Blaylock and Allen also report results on predicting arguments, but using a methodology that makes a direct comparison difficult.

	Monroe	Linux
Blaylock	94.20	36.10
MLN-manual	90.80	16.19
MLN-HCAM	97.00	38.94
BALP	98.40	46.60

Table 2: Convergence scores for Monroe and Linux datasets

BALPs and MLN-HCAM recognize plans in these domains more accurately than Blaylock and Allen's system. The performance of BALP and MLN-HCAM are fairly comparable on Monroe; however BALPs are significantly more accurate on Linux. The traditional MLN performs the worst, and does particularly poorly on Linux. This demonstrates the value of the abductive approach as implemented in MLN-HCAM.

Partial Observability Results

The convergence score has the following limitations as a metric for evaluating the performance of plan recognition:

1. It only accounts for predicting the correct plan predicate, ignoring the arguments. In most domains, it is important for a plan-recognition system to accurately predict arguments as well. For example, in the Linux domain, if the user is trying to move "test1.txt" to "test-dir", it is not sufficient to just predict the move command; it is also important to predict the file (test.txt) and the destination directory (test-dir).
2. It only evaluates plan prediction after the system has observed *all* of the executed actions. However, in most cases, we would like to be able to predict plans after observing as few actions as possible.

In order to evaluate the ability of BALPs and MLNs to infer plan arguments and to predict plans after observing only a partial execution, we conducted an additional set of experiments. Specifically, we performed plan recognition after observing the first 25%, 50%, 75%, and 100% of the executed actions. To measure performance, we compared the complete inferred plan (with arguments) to the gold-standard to compute an overall *accuracy* score. As for Story Understanding,

	25%	50%	75%	100%
MLN-manual	10.63	13.23	37.00	67.13
MLN-HCAM	15.93	19.93	43.93	76.30
BALP	07.33	20.26	44.63	79.16

Table 3: Accuracy on Monroe data for varying levels of observability

	25%	50%	75%	100%
MLN-manual	10.61	10.72	10.61	10.61
MLN-HCAM	16.30	16.48	24.36	28.84
BALPs	19.83	25.45	34.06	36.32

Table 4: Accuracy on Linux data for varying levels of observability

partial credit was given for predicting the correct plan predicate with only a subset of its correct arguments.

Table 3 shows the results for partial observability for the Monroe data. BALPs performs slightly better than MLN-HCAM on higher levels of observability, whereas, MLN-HCAM tends to outperform BALP on lower levels of observability. The MLN-manual performs worst at higher levels of observability, but at 25% observability, it out-performs BALPs. Table 4 shows the results for partial observability on the Linux data. Here, BALPs clearly outperform MLN-HCAM and traditional MLNs at all the levels of observability. The traditional MLN performs significantly worse than the other two models, especially at higher levels of observability. For Story Understanding, since the set of observed actions is already incomplete, we did not perform additional experiments for partial observability.

5.2.3. Discussion

We believe several aspects of SRL approaches led to their superior performance over existing approaches like ACCEL and Blaylock and Allen's system. The ability of BALPs and MLN-HCAM to perform probabilistic reasoning most likely resulted in their improved performance over ACCEL-Simplicity, a standard logical abduction method. When Blaylock and Allen (2005) 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 and MLN-HCAM are able to jointly predict both the plan schema and its arguments simultaneously. We believe that this ability of these SRL models to perform joint prediction of plans and their arguments is at least partially responsible for their superior performance. In addition, both BALPs and MLN-HCAM utilize prior planning knowledge encoded in the logical clauses given to the system, while Blaylock and Allen's system has no access to such planning knowledge. We believe that the ability of SRL models to incorporate such prior domain knowledge also contributes to their superior performance.

MLN-manual, although a joint model, cannot take advantage of all of the domain knowledge in the planning KB available to BALPs and MLN-HCAM. Also, it does not have the advantages offered by the abductive model construction process used in MLN-HCAM. This also makes it difficult to adequately learn the parameters for this model. We believe these factors lead to its overall inferior performance compared to the other models. For both the Monroe and Linux domains, the relative gap in the performance of MLN-manual model decreases with decreasing observability. This is particularly evident in Linux, where the performance stays almost constant with decreasing observability. We believe this is due to the model's ability to capitalize on even a small amount of information that deterministically predicts the top-level plan. Furthermore, at lower levels of observability, the ground networks are smaller and therefore, approximate inference is more likely to be accurate. Singla and Mooney (2011) report that MLN-PC and MLN-HC models did not scale well enough to make them tractable for the Monroe and Linux domains. When compared to these models, MLN-manual has a substantial advantage. But it still does not perform nearly as well as the MLN-HCAM model. This re-emphasizes the importance of using a model that is constructed by focusing on both the query and the available evidence. Furthermore, the MLN-manual approach requires costly human labor to engineer the knowledge base. This is in contrast to the abductive MLN models that allow the same knowledge base to be used for both planning *and* plan recognition.

Comparing BALPs and MLN-HCAM, BALPs generally performed better. We believe this difference is partly due to the advantages that directed graphical models have for abductive reasoning, as originally discussed by Pearl (1988). Note that MLN-HCAM already incorporates several ideas that originated with BALPs. By using hidden causes and noisy-or to combine evidence from multiple rules, and by utilizing logical abduction to obtain a focused set of literals for the ground net-

work, we improved the performance of MLNs by making them produce a graphical model that is more similar to the one produced by BALPs. Although, in principle, any directed model can be re-expressed as an undirected model, the learning of parameters in the corresponding undirected model can be significantly more complex since there is no closed form solution for the maximum-likelihood set of parameters unlike in the case of directed models [5] (Koller and Friedman, 2009). Inaccurate learning of parameters can lead to potential loss of accuracy during final classification. Undirected models do have the advantage of representing cyclic dependencies (such as transitivity), which directed models can't represent explicitly because of the acyclicity constraint. But we did not find it particularly useful for plan recognition since the domain knowledge is expressed using rules that have an inherent directional (causal) semantics. In addition, it is very difficult to develop a general MLN method that constructs a Markov net that is formally equivalent to the Bayes net constructed by a BALP given the same initial planning knowledge base.

In general, it took much more engineering time and effort to get MLNs to perform well on plan recognition compared to BLPs. Extending BLPs with logical abduction was straightforward. The main problem we encountered while adapting BLPs to work well on our plan recognition benchmarks was finding an effective approximate inference algorithm that scaled well to the larger Monroe and Linux datasets. Once we switched to Sample Search, which is designed to work well with the mix of soft and hard constraints present in our networks, BALPs produced good results. However, getting competitive results with MLNs and scaling them to work with our larger datasets was significantly more difficult. First, we needed to develop a method for properly introducing reverse clauses to allow MLNs to perform logical abduction. Next, we had to develop a method for introducing hidden causes in order to prevent the creation of networks with large cliques that made inference intractable. Finally, we had to develop an abductive model construction process that used the ground literals constructed for BALPs to constrain the size and scope of the ground Markov net. Even after all these modifications, the weight-learning algorithm did not scale to larger training sets, and the overall results are still not competitive with BALPs.

[5] In directed models, a closed form solution exists for the case of full observability. This corresponds to the M step in EM when dealing with partial observability.

Some of the differences in the performance of BALPs and MLN-HCAM may also stem from the differences in the probabilistic inference and parameter-learning algorithms employed. For instance, on the Linux dataset, we could run exact inference for BALPs, but we had to resort to MC-SAT, an approximate sampling algorithm, for MLN-HCAM. On Monroe, even though we used approximate sampling algorithms for both BALPs and MLN-HCAM, it is unclear whether the performance of Sample Search and MC-SAT are comparable. Furthermore, since probabilistic inference is used extensively during parameter learning, performance of the inference techniques could impact the quality of the learned weights/parameters. In our preliminary work, we converted the noisy-or parameters learned using the EM algorithm for BALPs into weights in MLN-HCAM. When we performed plan recognition using these weights, we found that the performance of MLN-HCAM improved, demonstrating a lack of quality in the learned MLN weights. This could be due either to poor performance of probabilistic inference, or to poor performance of the weight learner itself. Additional experiments that control for changes in the inference and learning algorithms are needed to further understand the effects of these differences.

5.3. Comparison of BALPs and MLNs to other SRL models

BLPs, BALPs, and MLNs are all languages for flexibly and compactly representing large, complex probabilistic graphical models. An alternative approach to SRL is to add a stochastic element to the deductive process of a logic program. ProbLog (Kimmig et al., 2008), is the most recent and well-developed of these approaches. ProbLog can be seen as extending and subsuming several previous models, such as Poole's Probabilistic Horn Abduction (PHA) (Poole, 1993) and PRISM (Sato, 1995). Finally, there is publicly-available implementation of ProbLog [6] that exploits the latest inference techniques based on *binary decision diagrams* (BDDs) to provide scalability and efficiency.

Therefore, we attempted to also compare the performance of our models to ProbLog. It was relatively straightforward to develop a ProbLog program for plan-recognition by appropriately formulating the planning KB used for both BLPs and abductive MLNs. However, our preliminary explorations with ProbLog revealed a serious

[6] `http://dtai.cs.kuleuven.be/problog/`

limitation that prevented us from actually performing an experimental comparison on our plan recognition datasets. In a number of the planning axioms in our KBs, existentially quantified variables occur in the body of a clause which do not occur in the head. Representing these clauses in ProbLog requires binding such variables to all possible type-consistent constants in the domain. However, this results in the ProbLog inference engine attempting to construct an intractable number of explanations (i.e. proofs) due to the combinatorial number of possible combinations of these introduced constants. Therefore, it was intractable to run ProbLog on our datasets, preventing an empirical comparison. BALPs and MLN-HCAM use a greedy abductive-proof construction method described in section 3.1 to prevent this combinatorial explosion. Therefore, we believe ProbLog would need a new approximate inference algorithm for this situation in order to be practically useful for plan recognition.

Abductive Stochastic Logic Programs (ASLPs) (Chen et al., 2008) are another SRL model that uses stochastic deduction and supports logical abduction and, therefore, could potentially be applied to plan recognition. However, we are unaware of a publicly-available implementation of ASLPs that could be easily used for experimental comparisons.

6. Future Work

The research presented in this paper could be extended in various ways. First, it would be good to evaluate the proposed plan-recognition systems on additional domains and applications. Unfortunately, there are very few publicly-available datasets for plan recognition.

Second, the existing SRL methods could be improved and extended in several productive directions. Methods for lifted inference (Singla and Domingos, 2008) could improve efficiency by allowing probabilistic inference to be performed without having to explicitly construct complete ground networks. In particular, the latest Probabilistic Theorem Proving (PTP) methods for lifted inference in MLNs (Gogate and Domingos, 2011) could be tried to improve the efficiency and accuracy of the MLN models.

Improved on-line weight learning algorithms could be developed to more effi-

ciently train on large datasets and increase the accuracy of the learned models. In particular, discriminative rather than generative (i.e. EM) parameter learning for BALPs should be explored. Although discriminative learning is more difficult for directed graphical models than for undirected ones, there has been recent progress on this problem (Carvalho et al., 2011). Current discriminative on-line weight learners for MLNs (Huynh and Mooney, 2011a) assume completely observable training data. These methods are not applicable to abductive MLNs, which contain unobserved sub-goal and noisy-or nodes. Therefore, effective on-line methods for partially-observed training data need to be developed.

With respect to the traditional MLN approach, better methods for manually engineering effective rules for deductive plan recognition could be developed. Alternatively, MLN structure learning (Kok and Domingos, 2005, 2010; Huynh and Mooney, 2011b) could be used to automatically induce such rules from supervised training data. In addition, a similar approach could be developed for applying traditional (deductive) BLPs to plan recognition.

The current experimental comparisons should be extended to additional SRL models. As mentioned in section 5.3, an improved approximate inference method is needed to make ProbLog tractable for our plan-recognition problems. Comparisons to other SRL models such as Poole's Horn Abduction (Poole, 1993), PRISM (Sato, 1995), and Abductive Stochastic Logic Programs (Chen et al., 2008), are also indicated.

7. Conclusions

This paper has introduced two new SRL approaches to plan recognition, one based on Bayesian Logic Programs (BLPs), the other on Markov Logic Networks (MLNs). Both of these approaches combine the advantages of prior logical and probabilistic methods. We presented novel techniques for extending both MLNs and BLPs with logical abduction in order to allow for plan recognition given logical definitions of actions and plans as the only prior knowledge. Experimental evaluations on three benchmark data sets have shown that our approaches generally outperform other state-of-the-art methods in plan recognition. We believe their superior performance is due to the combination of logical abduction, joint probabilistic inference, and incorporation of planning domain knowledge. The

results also indicate that the approach based on BLPs is generally more effective than the one based on MLNs.

Acknowledgements

We would like to thank Nate Blaylock for sharing the Linux and Monroe data sets and Vibhav Gogate for helping us modify SampleSearch for our experiments. We would also like to thank Luc De Raedt and Angelika Kimmig for their help in our attempt to run ProbLog on our plan recognition datasets. This research was funded by MURI ARO grant W911NF-08-1-0242 and 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. All views expressed are solely those of the authors and do not necessarily reflect the opinions of ARO, DARPA, NSF or any other government agency.

References

Albrecht DW, Zukerman I, Nicholson AE. Bayesian models for keyhole plan recognition in an adventure game. In: User Modeling and User-Adapted Interaction. 1998. p. 5–47.

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

Blaylock N, Allen JF. Statistical goal parameter recognition. In: Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS-04). 2004. p. 297–305.

Breese JS, Goldman RP, Wellman MP, editors. IEEE Transactions on Systems, Man and Cybernetics (Vol 24, No. 11): Special Issue on Knowledge Based Construction of Probabilistic and Decision Models. IEEE Society Press, 1994.

Bui HH. A general model for online probabilistic plan recognition. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-2003). Acapulco, Mexico; 2003. p. 1309–15.

Bui HH, Venkatesh S, West G. Policy recognition in abstract hidden Markov model. Journal of Artificial Intelligence Research 2002;17:451–99.

Carvalho AM, Roos T, Oliveira AL, Myllymäki P. Discriminative learning of Bayesian networks via factorized conditional log-likelihood. Journal of Machine Learning Research 2011;12:2181–210.

Charniak E, Goldman R. A probabilistic model of plan recognition. In: Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91). Anaheim, CA; 1991. p. 160–5.

Charniak E, Goldman RP. 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; 1989. p. 1074–9.

Charniak E, Goldman RP. Plan recognition in stories and in life. In: Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence. UAI 1989; 1990. p. 343–52.

Charniak E, McDermott D. Introduction to Artificial Intelligence. Reading, MA: Addison-Wesley, 1985.

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

Davis J, Ong I, Struyf J, Costa VS, Burnside E, Page D. Change of representation for statistical relational learning. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-2007). Hyderabad, India; 2007. p. 2719–26.

Domingos P, Lowd D. Markov Logic: An Interface Layer for Artificial Intelligence. San Rafael, CA: Morgan & Claypool, 2009.

Elvira-Consortium . ELVIRA: An environment for probabilistic graphical models. In: Proceedings of the Workshop on Probabilistic Graphical Models. Cuenca, Spain; 2002. p. 222–30.

Getoor L, Taskar B, editors. Introduction to Statistical Relational Learning. Cambridge, MA: MIT Press, 2007.

Gogate V, Dechter R. Samplesearch: Importance sampling in presence of determinism. Artificial Intelligence 2011;175:694–729.

Gogate V, Domingos P. Probabilistic theorem proving. In: Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11). Corvallis, Oregon: AUAI Press; 2011. p. 256–65.

Goldman RP, Geib CW, Miller CA. A new model for plan recognition. In: Proceedings of 15th Conference on Uncertainty in Artificial Intelligence (UAI-99). 1999. p. 245–54.

Hobbs JR, Stickel ME, Martin P, Edwards D. Interpretation as abduction. In: Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics (ACL-88). Buffalo, New York; 1988. p. 95–103.

Hong J. Goal recognition through goal graph analysis. Journal of Artificial Intelligence Research 2001;15:1–30.

Horvitz E, Paek T. A computational architecture for conversation. In: Proceedings of the Seventh International Conference on User Modeling. Springer; 1999. p. 201–10.

Huber MJ, Durfee EH, Wellman MP. The automated mapping of plans for plan recognition. In: Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann; 1994. p. 344–51.

Huynh TN, Mooney RJ. Online max-margin weight learning for Markov logic networks. In: Proceedings of the Eleventh SIAM International Conference on Data Mining (SDM-11). Mesa, AZ; 2011a. p. 642–51.

Huynh TN, Mooney RJ. Online structure 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-11). Athens, Greece; 2011b. p. 81–96.

Inoue N, Ovchinnikova E, Inui K, Hobbs J. Weighted abduction for discourse processing based on integer linear programming. In: Sukthankar G, Goldman R, Geib C, Pynadath D, Bui H, editors. Plan, Activity, and Intent Recognition: Theory and Practice. Elsevier; 2013. .

Kakas AC, Kowalski RA, Toni F. Abductive logic programming. Journal of Logic and Computation 1993;2(6):719–70.

Kaminka GA, Pynadath DV, Tambe M. Monitoring teams by overhearing: A mulit-agent plan-recognition approach. Journal of Artificial Intelligence Research 2002;17:83–135.

Kate RJ, Mooney RJ. Probabilistic abduction using Markov logic networks. In: Proceedings of the IJCAI-09 Workshop on Plan, Activity, and Intent Recognition (PAIR-09). Pasadena, CA; 2009. .

Kautz HA. A Formal Theory of Plan Recognition. Ph.D. thesis; Department of Computer Science, University of Rochester; Rochester, NY; 1987. Technical Report 215.

Kautz HA, Allen JF. Generalized plan recognition. In: Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86). Philadelphia, PA; 1986. p. 32–7.

Kersting K, De Raedt L. Towards combining inductive logic programming with Bayesian networks. In: Proceedings of the 11th International Conference on Inductive Logic Programming (ILP-2001). Strasbourg, France; 2001. p. 118–31.

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

Kersting K, De Raedt L. Basic principles of learning Bayesian logic programs. In: Probabilistic Inductive Logic Programming. 2008. p. 189–221.

Kimmig A, Santos Costa V, Rocha R, Demoen B, De Raedt L. On the efficient execution of ProbLog programs. In: Proceedings of the 24th International Conference on Logic Programming (ICLP-08). Berlin, Heidelberg: Springer-Verlag; 2008. p. 175–89.

Kok S, Domingos P. Learning the structure of Markov logic networks. In: Proceedings of 22nd International Conference on Machine Learning (ICML-2005). Bonn,Germany; 2005. p. 441–8.

Kok S, Domingos P. Statistical predicate invention. In: Proceedings of 24th International Conference on Machine Learning (ICML-2007). Corvallis,OR; 2007. p. 433–40.

Kok S, Domingos P. Learning Markov logic networks using structural motifs. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10). Haifa, Israel; 2010. p. 551–8.

Kok S, Sumner M, Richardson M, Singla P, Poon H, Lowd D, Wang J, Nath A, Domingos P. The Alchemy System for Statistical Relational AI. Technical Report; Department of Computer Science and Engineering, University of Washington; 2010.

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

Lesh N, Etzioni O. A sound and fast goal recognizer. In: Proceedings of Fourteenth International Joint Conference on Artificial Intelligence (IJCAI). Morgan Kaufmann; 1995. p. 1704–10.

Levesque HJ. A knowledge-level account of abduction. In: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89). Detroit, MI; 1989. p. 1061–7.

Natarajan S, Khot T, Kersting K, Tadepalli P, Shavlik J. Exploiting causal independence in Markov logic networks: Combining undirected and directed models. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-10). Barcelona, Spain; 2010. p. 434–50.

Ng HT, Mooney RJ. The role of coherence in abductive explanation. In: Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90). Detroit, MI; 1990. p. 337–442.

Ng HT, Mooney RJ. Abductive plan recognition and diagnosis: A comprehensive empirical evaluation. In: Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning. Cambridge, MA; 1992. p. 499–508.

Ngo L, Haddawy P. Answering queries from context-sensitive probabilistic knowledge bases. Theoretical Computer Science 1997;171:147–77.

Nilsson D. An efficient algorithm for finding the M most probable configurations in probabilistic expert systems. Statistics and Computing 1998;8:159–73.

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

Peng Y, Reggia JA. Abductive Inference Models for Diagnostic Problem-Solving. New York: Springer Verlag, 1990.

Poole D. Probabilistic Horn abduction and Bayesian networks. Artificial Intelligence 1993;64:81–129.

Poon H, Domingos P. Sound and efficient inference with probabilistic and deterministic dependencies. In: Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06). Boston, MA; 2006. p. 458–63.

Pople HE. On the mechanization of abductive logic. In: Proceedings of the Third International Joint Conference on Artificial Intelligence (IJCAI-73). 1973. p. 147–52.

Pynadath DV, Wellman MP. Probabilistic state-dependent grammars for plan recognition. In: In Proceedings of the Conference on Uncertainty in Artificial Intelligence, UAI2000. Morgan Kaufmann Publishers; 2000. p. 507–14.

Raghavan S, Mooney RJ. Abductive plan recognition by extending Bayesian logic programs. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-11). Athens, Greece; 2011. p. 629–44.

Richardson M, Domingos P. Markov logic networks. Machine Learning 2006;62:107–36.

Riedel S. Improving the accuracy and efficiency of MAP inference for Markov logic. In: Proceedings of 24th Conference on Uncertainty in Artificial Intelligence (UAI-2008). Helsinki, Findland; 2008. p. 468–75.

Sadilek A, Kautz H. Modeling and reasoning about success, failure, intent of multi-agent activities. In: Proceedings of the UbiComp 2010 Workshop on Mobile Context-Awareness. Copenhagen, Denmark; 2010a. .

Sadilek A, Kautz H. Recognizing multi-agent activities from GPS data. In: Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-10). Atlanta, GA; 2010b. p. 1134–9.

Saria S, Mahadevan S. Probabilistic plan recognition in multiagent systems. In: International Conference on Automated Planning and Scheduling (ICAPS 2004). 2004. p. 287–96.

Sato T. A statistical learning method for logic programs with distribution semantics. In: Proceedings of the Twelfth International Conference on Logic Programming (ICLP-95). MIT Press; 1995. p. 715–29.

Shavlik J, Natarajan S. Speeding up inference in Markov logic networks by preprocessing to reduce the size of the resulting grounded network. In: Proceedings of the Twenty First International Joint Conference on Artificial Intelligence (IJCAI-2009). Hyederabad, India; 2009. p. 1951–6.

Singla P, Domingos P. Discriminative training of Markov logic networks. In: Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05). Pittsburgh, PA; 2005. p. 868–73.

Singla P, Domingos P. Lifted first-order belief propagation. In: Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI-08). Chicago, IL; 2008. p. 1094–9.

Singla P, Mooney R. Abductive Markov logic for plan recognition. In: Twenty-fifth National Conference on Artificial Intelligence. 2011. p. 1069–75.

Stickel ME. A Prolog-like Inference System for Computing Minimum-Cost Abductive Explanations in Natural-Language Interpretation. Technical Report Technical Note 451; SRI International; Menlo Park, CA; 1988.

Wellman MP, Breese JS, Goldman RP. From knowledge bases to decision models. The Knowledge Engineering Review 1992;7(01):35–53.