Combining Relational Learning with SMT Solvers using CEGAR

Arun Chaganty[1], Akash Lal[2], Aditya V. Nori[2], and Sriram K. Rajamani[2]

[1] Stanford / `chaganty@stanford.com`
[2] Microsoft Research, India / {`akashl, adityan, sriram`}`@microsoft.com`

Abstract. In statistical relational learning, one is concerned with inferring the most likely explanation (or *world*) that satisfies a given set of weighted constraints. The weight of a constraint signifies our confidence in the constraint, and the most likely world that explains a set of constraints is simply a satisfying assignment that maximizes the weights of satisfied constraints. The relational learning community has developed specialized solvers (e.g., Alchemy and Tuffy) for such weighted constraints independently of the work on SMT solvers in the verification community. In this paper, we show how to leverage SMT solvers to significantly improve the performance of relational solvers.
Constraints associated with a weight of 1 (or 0) are called *axioms* because they *must* be satisfied (or violated) by the final assignment. Axioms can create difficulties for relational solvers. We isolate the burden of axioms to SMT solvers and only lazily pass information back to the relational solver. This information can either be a subset of the axioms, or even *generalized* axioms (similar to predicate generalization in verification).
We implemented our algorithm in a tool called Soft-Cegar that outperforms state-of-the-art relational solvers Tuffy and Alchemy over four real-world applications. We hope this work opens the door for further collaboration between relational learning and SMT solvers.

1 Introduction

We propose using automated techniques developed in the verification community to improve the efficiency of solving statistical relational learning problems. We first introduce the relational learning problem, review existing solutions, and then present our improvements.

Given data in the form of relations, relational learning involves inferring new relationships that are likely present in the data. For instance, suppose that we are given a set of bibliographic records downloaded from the Internet, and predicates BibAuthor and BibTitle that associate a bibliographic record with its authors and its title, respectively. Because there are variations in how different websites abbreviate author names or paper titles (in addition to spelling mistakes), it may not be immediately clear which records refer to the same paper. A relational learning question is to infer which records (or authors or titles) are the same.

Such problems naturally involve an interplay between logic and probability. Logic provides the tools to state our intuitions about how new relationships can be derived from existing relationships. For example, we can represent the statement "if two papers have the same authors and the same title, then the two papers are the same" in logic using a formula F as follows:

$$\begin{aligned}&\forall a_0 a_1 b_0 b_1 t_0 t_1 . (\mathsf{SameAuthor}(a_0, a_1) \wedge \mathsf{BibAuthor}(a_0, b_0) \wedge \ \mathsf{BibAuthor}(a_1, b_1)\\ &\quad \wedge \mathsf{SameTitle}(t_0, t_1) \wedge \ \mathsf{BibTitle}(t_0, b_0) \wedge \mathsf{BibTitle}(t_1, b_1)) \Rightarrow \mathsf{SameBib}(b_0, b_1)\end{aligned}$$

where $\mathsf{SameAuthor}, \mathsf{SameTitle}$ and $\mathsf{SameBib}$ are the relations that we want to infer. Probability, on the other hand, provides the tools to deal with incompleteness of our models, uncertainty in the world, and errors in the data. Weighted formulae combine both logic and probability. An example of a weighted formula is $0.7 : F$, where the weight 0.7 denotes our confidence in F, i.e., it is our estimate of the probability with which a world satisfies F.

Existing state-of-the-art relational solvers are based on propositional logic. Thus, "structural" constraints such as the fact that $\mathsf{SameBib}$ must be an equivalence relation, are encoded using constraints with weight 1 (i.e., they must be satisfied in any world):

$$\begin{array}{ll}\text{Reflexivity: } \forall b_0 . & \mathsf{SameBib}(b_0, b_0)\\ \text{Symmetry: } \forall b_0 b_1 . & \mathsf{SameBib}(b_0, b_1) \Rightarrow \mathsf{SameBib}(b_1, b_0)\\ \text{Transitivity: } \forall b_0 b_1 b_2 . & \mathsf{SameBib}(b_0, b_1) \wedge \mathsf{SameBib}(b_1, b_2)\\ & \Rightarrow \mathsf{SameBib}(b_0, b_2)\end{array} \tag{1}$$

Similarly, the constraint that $\mathsf{BibAuthor}$ and $\mathsf{BibTitle}$ must encode functions that associate the same author and title with each paper, is specified as follows:

$$\begin{array}{ll}\forall a_0 a_1 b_0 b_1 . & (\mathsf{SameBib}(b_0, b_1) \wedge \mathsf{BibAuthor}(a_0, b_0) \wedge \mathsf{BibAuthor}(a_1, b_1))\\ & \Rightarrow \ \mathsf{SameAuthor}(a_0, a_1)\\ \forall t_0 t_1 b_0 b_1 . & (\mathsf{SameBib}(b_0, b_1) \wedge \mathsf{BibTitle}(t_0, b_0) \wedge \mathsf{BibTitle}(t_1, b_1))\\ & \Rightarrow \ \mathsf{SameTitle}(t_0, t_1)\end{array} \tag{2}$$

These constraints affect the scalability of relational solvers in two ways: First, since variables in these formulae are usually universally quantified, and existing relational solvers use propositional logic, these constraints have to be *grounded* [12,16] by instantiating the quantifiers over all constants in the dataset, resulting in many constraints. Second, these are *hard* constraints that must be satisfied, whereas the strength of relational solvers is in dealing with *soft* constraints (because the solvers are optimized to quickly find a good approximation to the ideal result in which violation of some soft constraints is acceptable).

On the other hand, SMT solvers fit the job for precisely solving hard constraints. Moreover, they offer specialized theories that may already capture some of the constraints implicitly. For instance, the fact that $\mathsf{SameBib}$ should be an equivalence relation can be captured by defining it as $\mathsf{SameBib}(b_0, b_1) \equiv (f(b_0) = f(b_1))$, where f is some uninterpreted function and "=" is the interpreted equality relation. By leveraging the theory of uninterpreted functions with equality, one can completely elide away the constraints of Formula 1 (and similarly for Formula 2) when using SMT solvers.

Our algorithm proceeds as follows. Let $\mathcal{F}$ be a set of weighted constraints such that $\mathcal{F} = \mathcal{A} \cup \mathcal{F}_s$, where $\mathcal{A}$ is the set of axioms. Let $\mathcal{F}_0 = \mathcal{F}_s$. Inspired by CEGAR (counterexample-guided abstraction refinement), we start by invoking an underlying relational learner (like TUFFY or ALCHEMY) with the set of formulae $\mathcal{F}_0$. Suppose the relational learner returns a world ω_0. We then check which axioms in $\mathcal{A}$ are violated by ω_0 using SMT solvers, and selectively instantiate axioms on the values of the relations from ω_0 which witness these violations, and add these axioms to $\mathcal{F}_0$ resulting in a larger set of formulae $\mathcal{F}_1$. Next, we invoke the relational learner again with the larger set of formulae $\mathcal{F}_1$, and the iterative process continues until we obtain a world $\hat{\omega}$ that satisfies all the axioms.

While CEGAR is very familiar to the program verification community, and has been used extensively in model checking [1,2,6] and in SMT solvers based on DPLL(T) [7,8], our use of CEGAR for relational learning is new, and requires overcoming several technical challenges.

First, we need to prove that lazily adding axioms does not affect the optimality of the relational learning solution. A key insight here is that satisfied axioms do not contribute to the weight assigned to a world, whereas soft formulae do (we formalize this in Section 2). As a result, if a world ω is optimal for a set of constraints $\mathcal{F}$ and ω happens to satisfy all the axioms in $\mathcal{A}$, then ω is also an optimal world for the set of constraints $\mathcal{F} \cup \mathcal{A}$.

Second, we find that the iterative CEGAR process sometimes requires a large number of iterations, each of which adds formulae forming particular patterns. We propose a technique to detect these patterns and suitably *generalize* the axioms that we add during refinement, thereby greatly reducing the number of iterations needed for convergence.

We have implemented our relational learning algorithm in a tool called SOFT-CEGAR and evaluated it on well-known applications. We show that SOFT-CEGAR outperforms state-of-the-art statistical inference tools such as TUFFY [16] and ALCHEMY [12], both in terms of efficiency and quality of results.

The rest of the paper is organized as follows. Section 2 describes preliminaries and formally defines the problem. Section 3 defines the SOFT-CEGAR algorithm and proves its correctness. Section 4 describes the empirical evaluation of SOFT-CEGAR on four applications. Section 5 surveys related work, and Section 6 concludes the paper.

2 Background: Statistical Relational Learning

We are interested in learning relations from a corpus of data given weighted formulae as specifications. The weight of a formula is a real number in the interval $[0, 1]$ that is used to model our confidence in the formula. The corpus of data is called *evidence*, which is an incomplete valuation of the relations. The goal of relational learning is to complete the valuation of the relations and infer a world in order to satisfy the specifications in an optimal manner.

The probabilistic models that we consider for relational learning problems are Markov Logic Networks [20].

2.1 Markov Logic Network

Definition 1. *A Markov Logic Network (MLN) $L = \langle \mathcal{D}, \mathcal{R}, \mathcal{F} \rangle$ is a triple, where*

- $\mathcal{D} = \{D_1, D_2, \ldots\}$ *is a set of finite domains.*
- $\mathcal{R} = \{R_1, R_2, \ldots\}$ *is a set of relations over these domains. We assume the existence of a function $\mathcal{S}$ that maps each relation in $\mathcal{R}$ to a schema. For instance, $\mathcal{S}(R_1)$ could be $D_1 \times D_3 \times D_5$, which specifies that R_1 is a three-column relation and $R_1 \subseteq D_1 \times D_3 \times D_5$.*
- $\mathcal{F}$ *is a set of weighted formulae of the form:* $\{w_1 : \forall \bar{x}_1.F_1(\bar{x}_1), w_2 : \forall \bar{x}_2.F_2(\bar{x}_2), \ldots, w_n : \forall \bar{x}_n.F_n(\bar{x}_n)\}$*, where each of the $w_i \in [0, 1]$ are real numbers, and each F_i is a formula in the Domain Relational Calculus (DRC) [22] over universally quantified variables $\bar{x}_i$ and the relations in set $\mathcal{R}$.*

We generalize the schema function $\mathcal{S}$ to both variables and formulas. Given a formula $f = w : \forall \bar{x}.F(\bar{x}) \in \mathcal{F}$, where $\bar{x} = x_1 \cdots x_n$, we define $\mathcal{S}(x_i, f)$ to be the domain of x_i in the formula, and $\mathcal{S}(F, f)$ to be $\mathcal{S}(x_1) \times \cdots \times \mathcal{S}(x_n)$. We will drop the argument f when it is clear from the context.
In the Domain Relational Calculus (DRC), every relation R is viewed as a predicate: $\forall \bar{c} \in \mathcal{S}(R)\ .\ R(\bar{c}) \Leftrightarrow \bar{c} \in R$. We now define the notion of formulae in a DRC. A term is either a constant c or variable x. Atoms are defined as follows:

- If R is a predicate with arity k and $t_1, \ldots, t_k$ are terms, then $R(t_1, \ldots, t_k)$ is an atom.
- If t_1 and t_2 are terms, then $t_1\ \Theta\ t_2$ is an atom, where $\Theta \in \{=, \neq\}$.

Next, we define formulae as follows.

- Every atom is a formula.
- If F_1 and F_2 are formulae, then so are $\neg F_1$, $F_1 \vee F_2$, $F_1 \wedge F_2$ and $F_1 \Rightarrow F_2$.

Figure 1 shows an example MLN $L = \langle \mathcal{D}, \mathcal{R}, \mathcal{F} \rangle$ for scheduling classes in a Computer Science department. Section (1) is the `Domain` section which defines the domains $\mathcal{D}$ or attributes of relations in the dataset. These are `Course` (set of courses offered), `Professor` (set of professors in the department), `Slot` (set of slots in a week) and `Student` (the set of students in the department).

Section (2) is the `Relations` section that defines the set of relations $\mathcal{R}$ of interest in the dataset. These are defined as follows.

- $\mathsf{Teaches}(p, c)$: Professor p teaches a course c.
- $\mathsf{Friends}(s_1, s_2)$: Student s_1 is a friend of student s_2.
- $\mathsf{Likes}(s, p)$: Student s likes professor p.
- $\mathsf{NextSlot}(s_1, s_2)$: Slot s_2 immediately follows slot s_1.
- $\mathsf{Attends}(s, c)$: Student s attends course c.
- $\mathsf{Popular}(p)$: Professor p is popular.
- $\mathsf{SameArea}(c_1, c_2)$: Courses c_1 and c_2 are in the same subarea of computer science.
- $\mathsf{HeldIn}(c, s)$: Course c is scheduled in slot s.

(1) Domains:
Course, Professor, Slot, Student
Course = {‘‘Algorithms and Complexity’’, ‘‘Medieval History of Machine Learning’’,...}
Professor = {‘‘Richard Karp’’, ‘‘C.A.R. Hoare’’, ...}
Slot = {‘‘Monday-Wednesday-Friday 9:00-10:00’’, ‘‘Tuesday-Thursday 9:00-10:30’’,... }
Student = { ‘‘Jay Leno’’, ‘‘David Letterman’’, ‘‘Bill Cosby’’, ... }

(2) Relations:
Teaches(Professor, Course), Friends(Student, Student), Likes(Student, Professor), NextSlot(Slot, Slot)
Attends(Student, Course), Popular(Professor), SameArea(Course, Course), HeldIn(Course, Slot)

(3) Weighted formulae:
(* Axiom: Professors and Students cannot be in two places at once *)
1.0: $\forall p_1 c_1 c_2 s_2.\mathsf{Teaches}(p_1, c_1) \wedge \mathsf{Teaches}(p_1, c_2) \wedge \mathsf{HeldIn}(c_1, s_1) \wedge \mathsf{HeldIn}(c_2, s_2) \wedge c_1 \neq c_2 \Rightarrow s_1 \neq s_2$
1.0: $\forall s_1 c_1 c_2 r_1 r_2.\mathsf{Attends}(s_1, c_1) \wedge \mathsf{Attends}(s_1, c_2) \wedge \mathsf{HeldIn}(c_1, r_1) \wedge \mathsf{HeldIn}(c_2, r_2) \wedge c_1 \neq c_2 \Rightarrow r_1 \neq r_2$

(* Axiom: SameArea is an equivalence relation *)
1.0: $\forall c_1 c_2.\mathsf{SameArea}(c_1, c_1)$
1.0: $\forall c_1 c_2.\mathsf{SameArea}(c_1, c_2) \Rightarrow \mathsf{SameArea}(c_2, c_1)$
1.0: $\forall c_1 c_2 c_3.\mathsf{SameArea}(c_1, c_2) \wedge \mathsf{SameArea}(c_2, c_3) \Rightarrow \mathsf{SameArea}(c_1, c_3)$

(* Axiom: Friends is a symmetric relation *)
1.0: $\forall s_1 s_2.\mathsf{Friends}(s_1, s_2) \Rightarrow \mathsf{Friends}(s_2, s_1)$

(* Soft formula: Prefer courses offered by professors you like, and those your friends are taking *)
$0.7 : \forall p_1 c_1 s_1 p_1.\mathsf{Teaches}(p_1, c_1) \wedge \mathsf{Likes}(s_1, p_1) \Rightarrow \mathsf{Attends}(s_1, c_1)$
$0.7 : \forall p_1 c_1 s_1.\mathsf{Teaches}(p_1, c_1) \wedge \mathsf{Popular}(p_1) \Rightarrow \mathsf{Attends}(s_1, c_1)$
$0.7 : \forall s_1 s_2 c_1.\mathsf{Friends}(s_1, s_2) \wedge \mathsf{Attends}(s_1, c_1) \Rightarrow \mathsf{Attends}(s_2, c_1)$

(* Soft formula: Take related courses in same area to gather expertise *)
$0.7 : \forall s_1 c_1 c_2.\mathsf{Attends}(s_1, c_1) \wedge \mathsf{SameArea}(c_1, c_2) \Rightarrow \mathsf{Attends}(s_1, c_2)$

(* Soft formula: Try to schedule classes students attend in consecutive slots
$0.7 : \forall s_1 c_1 c_2 r_1 r_2.\mathsf{Attends}(s_1, c_1) \wedge \mathsf{Attends}(s_1, c_2) \wedge c_1 \neq c_2 \wedge \mathsf{HeldIn}(c_1, r_1) \wedge \mathsf{HeldIn}(c_2, r_2) \Rightarrow \mathsf{NextSlot}(r_1, r_2) \vee \mathsf{NextSlot}(r_2, r_1)$

(* Soft formula: Students tend to take courses in the same area *)
$0.7 : \forall s_1 c_1 c_2.\mathsf{Attends}(s_1, c_1) \wedge \mathsf{Attends}(s_1, c_2) \Rightarrow \mathsf{SameArea}(c_1, c_2)$
$0.7 : \forall s_1 c_1 c_2.\mathsf{Teaches}(s_1, c_1) \wedge \mathsf{Teaches}(s_1, c_2) \Rightarrow \mathsf{SameArea}(c_1, c_2)$

(* Soft formula: Professors are popular if several students like them *)
$0.7 : \forall s_1 p_1.\mathsf{Likes}(s_1, p_1) \Rightarrow \mathsf{Popular}(p_1)$

(4) Evidence:
(* complete values for the relations Teaches, Friends, Likes, and NextSlot*)
Teaches(‘‘Richard Karp’’, ‘‘Algorithms and Complexity Theory’’)
...
Friends(‘‘Jay Leno’’, ‘‘David Letterman’’)
...
Likes(‘‘Jay Leno’’, ‘‘Richard Karp’’)
...
NextSlot(‘‘Monday-Wednesday-Friday 9:00-10:00’’, ‘‘Monday-Wednesday-Friday 10:00-11:00’’)
NextSlot(‘‘Tuesday-Thursday 9:00-10:30’’, ‘‘Tuesday-Thursday 10:30-12:00’’)
...

Fig. 1. An MLN for a Computer Science department together with evidence.

Section (3) is the `Weighted formulae` section where all the axioms and soft formulae in $\mathcal{F}$ are defined. Recall that axioms are formulae that must definitely hold. The first two axioms state that professors and students cannot be in two places at the same time. The next set of axioms encode the fact that the relation SameArea is an equivalence relation. In other words, SameArea is a reflexive, symmetric and transitive relation. Finally, we have an axiom that states that the relation Friends is a symmetric relation. The first set of soft formulae specify a student's preference for courses offered by professors that she likes and for the courses taken by her friends. All these formulae are associated with a weight or confidence of 0.7. The next soft formula states that it is likely that students take courses in the same area with the intention of gaining expertise in that area. We also have a soft formula that tries to schedule classes for a student in consecutive slots for the sake of convenience. We have a soft formula that groups two courses into the same area if there are many students who take both courses. Finally, we have a soft formula which says that professors are popular when there are many students who like them.

Section (4) is the `Evidence` section that specifies known relations. The evidence can be thought of as a hard constraint that fixes the values of certain relations. In our dataset, the relations Teaches, Friends, Likes and NextSlot are completely determined. In this example, we are interested in the HeldIn relation, which is a schedule that assigns a course to a slot.

An *axiom* is a weighted formula with weight $w \in \{0, 1\}$. Axioms represent formulae in an MLN that *must* be satisfied or violated. Let $\mathcal{A}(L)$ be the set of axioms in an MLN L.

A *world* of an MLN $L = \langle \mathcal{D}, \mathcal{R}, \mathcal{F} \rangle$ is an assignment of all relations in $\mathcal{R}$ according to their schemas. Given a world u and a formula f with weight w, let $\Phi(w, f)$ be w if u satisfies f, and $(1 - w)$ otherwise. The *weight* of a world w, which is an estimate of the likelihood of u is simply defined as: $\prod_{f \in \mathcal{F}} \Phi(u, f)$. The *Maximum a Posteriori Probability* estimate (MAP) is defined as the world with maximum weight. It is possible for an MLN to have multiple MAP solutions; we are only interested in finding one.

Remark. In the verification community, the more common optimization problem is MAXSAT where the objective is to maximize the *sum* of weights of satisfied constraints. If one replaces the weights in an MLN with their logarithms, then solving MAXSAT over such an MLN is similar to computing its MAP. In other words, it is possible to replace relational solvers with MAXSAT solvers. In this paper, however, we only augment relational solvers with SMT solvers. A full comparison of relational solvers to MAXSAT is an interesting direction that we leave as future work.

Under this definition of MAP, we note that negating a weighted formula $w : \forall \bar{x}. F(\bar{x})$ can be done in two ways:

1. Flip the weight, resulting in $f_1 = (1 - w) : \forall \bar{x}. F(\bar{x})$, or
2. Negate the formula, resulting in $f_2 = w : \forall \bar{x}. \neg F(\bar{x})$.

In other words, the formulae f_1 and f_2 are equivalent because using either does not change the MAP. Notice that the universal quantifier does not change to an existential quantifier.

Computing the MAP world of an MLN is $\mathsf{NP\text{-}hard}$. However, there are a number of machine learning techniques that efficiently estimate the MAP solution for an MLN given the evidence. For instance, the WalkSAT algorithm [9] that forms the basis of many statistical relational learning tools [12, 16] is one such technique.

The next section shows how we can exploit the distinction between axioms and other soft formulae in MLNs to make exact and approximate MAP estimation more scalable along with a potential for much improved precision.

3 The Soft-Cegar Algorithm

In this section, we describe an algorithm called Soft-Cegar for efficiently computing the MAP for an input MLN L. Soft-Cegar provides a framework for systematically combining any relational MAP inference algorithm with logical inference algorithms that check consistency of the MAP solutions with respect to the axioms defined by $\mathcal{A}(L)$.

We draw inspiration from the following key ideas in program verification and theorem proving: (a) Counterexample-guided abstraction refinement (CEGAR) [6], (b) theorem proving [7], and (c) generalization techniques for accelerating convergence of the CEGAR loop [13].

Notation. For an axiom $f = w : \forall\bar{x}.F(\bar{x})$, let $\llbracket f \rrbracket$ be $\forall\bar{x}.F(\bar{x})$ if $w = 1.0$ and $\forall\bar{x}.\neg F(\bar{x})$ if $w = 0.0$. For a set of axioms A, let $\llbracket A \rrbracket = \wedge\{\llbracket f \rrbracket \mid f \in A\}$. For two sets of axioms A_1 and A_2, we say that A_1 *entails* A_2, or $A_1 \models A_2$ if $(\llbracket A_1 \rrbracket \Rightarrow \llbracket A_2 \rrbracket)$ is valid. Note that if $A_1 = \{1.0 : \forall\bar{x}.F(\bar{x})\}$ and $A_2 = \{1.0 : F(\bar{c})\}$ for some $\bar{c} \in \mathcal{S}(\bar{x})$, then $A_1 \models A_2$.

The Soft-Cegar algorithm is described in Figure 2. The input to the algorithm is an MLN $L = \langle \mathcal{D}, \mathcal{R}, \mathcal{F} \rangle$ and its output is a world ω_{MAP} that is a MAP solution of L.

The CEGAR loop of the Soft-Cegar algorithm is described in lines 3–24. The set $\mathcal{F}_{\mathsf{approx}}$ contains all weighted constraints of the MLN, except for its axioms $\mathcal{A}(L)$. We use the set $\mathcal{C}$ to capture a subset of the axioms (or their ground instances). Initially, $\mathcal{C}$ is empty and it grows iteratively inside the CEGAR loop. The algorithm always maintains the invariant that $\mathcal{A}(L) \models \mathcal{C}$.

In line 4, an approximation L_{approx} to the input MLN L is constructed. Note that because $\mathcal{C}$ is entailed by $\mathcal{A}(L)$, L_{approx} always has fewer (or same) number of constraints compared to L. In the first iteration, L_{approx} is the input MLN L without any axioms. Next, in line 6, an off-the-shelf MAP solver $\mathsf{Solve}_{\mathsf{map}}(L_{\mathsf{approx}})$ is invoked on the approximated MLN L_{approx}. This results in a MAP world ω_{approx} of the MLN L_{approx}.

Lines 9–17 check whether ω_{approx} is consistent with the axioms $\mathcal{A}(L)$ of the input MLN L. The set of conflicting axioms, i.e., axioms that are not satisfied by ω_{approx}, are collected in $\mathcal{C}'$. In lines 10–16, for every axiom $w : \forall\bar{x}.F(\bar{x})$, we

```
algorithm SOFT-CEGAR

  input:
    $L = \langle \mathcal{D}, \mathcal{R}, \mathcal{F} \rangle$: MLN
  output:
    $\omega_{\mathsf{MAP}}$: MAP solution

 1: $\mathcal{F}_{\mathsf{approx}} := \mathcal{F} \setminus \mathcal{A}(L)$
 2: $\mathcal{C} := \emptyset$
 3: loop
 4:    $L_{\mathsf{approx}} := \langle \mathcal{D}, \mathcal{R}, \mathcal{F}_{\mathsf{approx}} \cup \mathcal{C} \rangle$
 5:    (* Call relational MAP solver *)
 6:    $\omega_{\mathsf{approx}} := \mathsf{Solve}_{\mathsf{map}}(L_{\mathsf{approx}})$
 7:    (* Find conflicting axioms *)
 8:    $\mathcal{C}' := \emptyset$
 9:    for each $w : \forall \bar{x}.F(\bar{x}) \in \mathcal{A}(L)$ do
10:       if $w = 1.0$ then
11:          $\mathcal{T} := \textsc{IsConsistent}(\omega_{\mathsf{approx}}, \neg F)$
12:          $\mathcal{C}' := \mathcal{C}' \cup \{1.0 : F(\bar{c}) \mid \bar{c} \in \mathcal{T}\}$
13:       else
14:          $\mathcal{T} := \textsc{IsConsistent}(\omega_{\mathsf{approx}}, F)$
15:          $\mathcal{C}' := \mathcal{C}' \cup \{0.0 : F(\bar{c}) \mid \bar{c} \in \mathcal{T}\})\}$
16:       end if
17:    end for
18:    if $\mathcal{C}' = \emptyset$ then
19:       $\omega_{\mathsf{MAP}} := \omega_{\mathsf{approx}}$
20:       return $\omega_{\mathsf{MAP}}$
21:    end if
22:    $\mathcal{C} := \mathcal{C} \cup \mathcal{C}'$
23:    $\mathcal{C} := \textsc{Generalize}(\mathcal{C}, L)$
24: end loop
```

Fig. 2. The SOFT-CEGAR algorithm.

```
algorithm GENERALIZE

  input:
    $\mathcal{C}$: a set of axioms $L$ : MLN
  parameters:
    $thresh$: a floating point number
  output:
    $\mathcal{G}$: a set of axioms such that $\mathcal{A}(L) \models \mathcal{G} \models \mathcal{C}$

 1: $\mathcal{G} := \emptyset$
 2: for all $\phi \in \mathcal{A}(L)$ do
 3:    let $\phi = 1.0 : \forall \bar{x} F(\bar{x})$
 4:    for all partial instantiations $\psi$ of $\phi$ do
 5:       let $covered := \{f \in \mathcal{C} \mid \{\psi\} \models \{f\}\}$
 6:       let $support := |covered| / |\mathcal{S}(F)|$
 7:       if $support > thresh$ then
 8:          (* Generalize *)
 9:          $\mathcal{G} := \mathcal{G} \cup \{\psi\}$
10:          $\mathcal{C} := \mathcal{C} \setminus covered$
11:          break
12:       end if
13:    end for
14: end for
15: return $\mathcal{G} \cup \mathcal{C}$
```

Fig. 3. The GENERALIZE algorithm.

compute a set of tuples $\mathcal{T} \subseteq \mathcal{S}(F)$ in ω_{approx} that violate the axiom (recall that $\mathcal{S}(F)$ is the product of the domains of the variables in F, as defined in Section 2). This is computed by the procedure ISCONSISTENT called on lines 11 and 14. This is essentially a call to an SMT solver. The set of conflict axioms $\mathcal{C}'$ is the set of all instances of the axiom $w : \forall \bar{x}.F(\bar{x})$ over the set of tuples $\mathcal{T}$. This is computed in lines 11 and 14.

If the set $\mathcal{C}'$ is empty, this means that the current solution ω_{approx} respects all axioms in $\mathcal{A}(L)$ and the procedure stops and returns $\omega_{\mathsf{MAP}} = \omega_{\mathsf{approx}}$ (line 20). Otherwise, the set of weighted formulae is refined with $\mathcal{C}'$ (line 22). It is easy to see that $\mathcal{A}(L) \models \mathcal{C}'$ because the latter only contains ground instances of axioms. Thus, when $\mathcal{C}$ is updated on line 22, we still have $\mathcal{A}(L) \models \mathcal{C}$.

In order to accelerate the convergence of the CEGAR loop, we make use of a generalization procedure (line 23). One can use any procedure as long as it satisfies the following condition: given argument $\mathcal{C}$, it must return a set A such that $\mathcal{A}(L) \models A \models \mathcal{C}$. Note that returning $\mathcal{C}$ itself is always a valid solution, but it does not accelerate convergence. Similarly, returning $\mathcal{A}(L)$ is also a valid solution, but this is too big a jump that may place a huge burden on the underlying solver. The particular implementation of generalization that we use chooses a middle ground. It is described in Figure 3 and the details will follow in Section 3.1.

Example. Consider the SameArea equivalence relation in Figure 1. Without the axiom of transitivity present in $\mathcal{F}_{\mathsf{approx}}$, a possible world of L_{approx} (on line 7) may have both the tuples SameArea("Static Analysis", "Program Analysis") and SameArea("Program Analysis", "Abstract Interpretation"), but not the tuple SameArea("Static Analysis", "Abstract Interpretation"). In this case, the set of tuples $\mathcal{T}$ (on line 10) returned by the SMT solver contains the tuple $\bar{c} =$ ("Static Analysis", "Program Analysis", "Abstract Interpretation"), and the conflict $F(\bar{c})$ added on line 11 is:
SameArea("Static Analysis", "Program Analysis") $\wedge$
SameArea("Program Analysis", "Abstract Interpretation")
$\Rightarrow$ SameArea("Program Analysis", "Abstract Interpretation")

This axiom prevents the MAP solver from choosing such a world in future iterations.

A property of the Soft-Cegar algorithm is that the laziness does not cause us to sacrifice precision. This is because satisfied axioms do not contribute to the weight assigned to a world. Thus, as long as all the axioms are satisfied, the weight of a world in an MLN with or without axioms is the same. We still need to establish that there is no other world that can satisfy all the axioms and have a higher weight. Assuming that the underlying relational learner is optimal, we can argue that such a situation cannot arise. Below, we exploit this natural separation that exists between soft formulas and axioms to show that the Soft-Cegar algorithm is able to compute an exact MAP solution, i.e., a world with the maximum weight.

Theorem 1. *The* Soft-Cegar *algorithm returns an exact MAP solution provided the MAP solver* $\mathsf{Solve}_{\mathsf{map}}$ *always returns exact MAP solutions.*

Proof: For simplicity, assume that axioms always have the weight 1.0 (otherwise, replace the axiom $0.0 : \forall \bar{x}.F(\bar{x})$ with $1.0 : \forall \bar{x}.\neg F(\bar{x})$). Recall the definition of the *weight* of a world from Section 2. Define the *weight* of an MLN to be the weight of its MAP solution, i.e., the maximum possible weight of a world in the MLN. Consider two MLNs $L_1 = \langle \mathcal{D}, \mathcal{R}, \mathcal{F} \cup \mathcal{C}_1 \rangle$ and $L_2 = \langle \mathcal{D}, \mathcal{R}, \mathcal{F} \cup \mathcal{C}_2 \rangle$ such that all the formulae in $\mathcal{C}_1$ and $\mathcal{C}_2$ are axioms with weight 1.0 and $\mathcal{C}_2 \models \mathcal{C}_1$. Then *weight*$(L_1) \geq$ *weight*(L_2) because:

A. Let ω be a world with weight $w \neq 0$ in L_2. Then ω must satisfy $\mathcal{C}_2$. Because $\mathcal{C}_2$ entails $\mathcal{C}_1$, ω satisfies $\mathcal{C}_1$. Because L_1 and L_2 have the same set of weighted constraints (barring axioms), the weight of ω in L_1 must also be w.
B. Let u be a MAP of L_2, i.e., its weight in L_2 is *weight*(L_2). From A, the weight of u in L_1 is also *weight*(L_2). (Nothing to prove if *weight*$(L_2) = 0$.)

C. If there is a world with weight w in L_1 then, by definition, $\mathit{weight}(L_1) \geq w$. Thus, from B, $\mathit{weight}(L_1) \geq \mathit{weight}(L_2)$.

Next, if u' is a MAP of L_1 and u' satisfies $\mathcal{C}_2$, then the weight of u' in L_2 is $\mathit{weight}(L_1)$. This implies $\mathit{weight}(L_1) = \mathit{weight}(L_2)$ and u' is a MAP of L_2.

The proof of the theorem follows from these observations. Let L_1 be the MLN L_{approx} on the last iteration of the SOFT-CEGAR algorithm. Then L_1 is $\langle \mathcal{D}, \mathcal{R}, \mathcal{F}_{\mathsf{approx}} \cup \mathcal{C} \rangle$ for some $\mathcal{C}$ that is accumulated in the various iterations of the loop. And the input MLN L can be written as $L_2 = \langle \mathcal{D}, \mathcal{R}, \mathcal{F}_{\mathsf{approx}} \cup \mathcal{A}(L) \rangle$. Because $\mathcal{A}(L) \models \mathcal{C}$, when $\mathcal{C}' = \emptyset$ on line 20 we know that ω_{approx} satisfies all the axioms $\mathcal{A}(L)$. In this case, the MAP solution of L_1 must be an MAP solution of L_2. ■

Theorem 1 assumes that the underlying MAP solver is exact. In practice, existing MAP solvers are based on probabilistic and approximation algorithms and cannot handle axioms precisely. As we show in the next section, even with these imprecise MAP solvers, SOFT-CEGAR's ability to handle axioms specially allows it to improve both runtime and precision of the inference.

3.1 Generalization

In this section, we describe the generalization procedure GENERALIZE employed by SOFT-CEGAR. As in program verification, the goal of generalization is to reduce the number of CEGAR iterations needed, while not imposing a huge burden on the underlying solver. We first work through an example.

Consider the course scheduling MLN from Figure 1. We assume that the following facts are part of the world ω_{approx} in some arbitrary iteration i of SOFT-CEGAR.

Attends(Student1, Course1)	HeldIn(Course1, Slot1)
Attends(Student1, Course3)	HeldIn(Course3, Slot1)

This world violates the following axiom that states that students cannot be in two places at the same time.

$$\begin{aligned} 1.0 : \ & \forall s_1 c_1 c_2 r_1 r_2 . \mathsf{Attends}(s_1, c_1) \wedge \mathsf{Attends}(s_1, c_2) \wedge \\ & \mathsf{HeldIn}(c_1, r_1) \wedge \mathsf{HeldIn}(c_2, r_2) \wedge c_1 \neq c_2 \Rightarrow r_1 \neq r_2 \end{aligned}$$

In order, to rule out this world ω_{approx}, the following conflict axiom is added to the set of weighted formulae for the approximate MLN in next iteration of SOFT-CEGAR.

$$\begin{aligned} 1.0 : \ & \neg\mathsf{Attends}(\text{Student1}, \text{Course1}) \vee \neg\mathsf{Attends}(\text{Student1}, \text{Course3}) \vee \\ & \neg\mathsf{HeldIn}(\text{Course1}, \text{Slot1}) \vee \neg\mathsf{HeldIn}(\text{Course3}, \text{Slot1}) \end{aligned}$$

However, resolving conflicts at such a fine granularity might be lead to a prohibitively large number of iterations and as a result, the following facts that violate the same axiom above might show up in worlds computed by subsequent iterations of SOFT-CEGAR.

i	$\mathsf{Attends}(\text{Student1}, \text{Course1})$ $\mathsf{HeldIn}(\text{Course1}, \text{Slot1})$	$\mathsf{Attends}(\text{Student1}, \text{Course3})$ $\mathsf{HeldIn}(\text{Course3}, \text{Slot1})$
$i+1$	$\mathsf{Attends}(\text{Student2}, \text{Course1})$ $\mathsf{HeldIn}(\text{Course1}, \text{Slot1})$	$\mathsf{Attends}(\text{Student2}, \text{Course3})$ $\mathsf{HeldIn}(\text{Course3}, \text{Slot1})$
$i+2$	$\mathsf{Attends}(\text{Student3}, \text{Course1})$ $\mathsf{HeldIn}(\text{Course1}, \text{Slot1})$	$\mathsf{Attends}(\text{Student3}, \text{Course3})$ $\mathsf{HeldIn}(\text{Course3}, \text{Slot1})$
$i+3$	$\mathsf{Attends}(\text{Student4}, \text{Course1})$ $\mathsf{HeldIn}(\text{Course1}, \text{Slot1})$	$\mathsf{Attends}(\text{Student4}, \text{Course3})$ $\mathsf{HeldIn}(\text{Course3}, \text{Slot1})$

Furthermore, it is possible that this sequence of conflict axioms could continue for many iterations. There could be several reasons for this behavior such as the popularity of Course1 and Course3. Therefore, we would like to find a more "general" conflict axiom that entails many possible conflict axioms in future iterations so as to reduce the overall number of iterations of Soft-Cegar. For instance, each of the following two axioms rule out all of the violating facts mentioned before.

$1.0 : \forall x. \neg\mathsf{Attends}(x, \text{Course1}) \vee \neg\mathsf{Attends}(x, \text{Course3}) \vee \neg\mathsf{HeldIn}(\text{Course1}, \text{Slot1}) \vee \neg\mathsf{HeldIn}(\text{Course3}, \text{Slot1})$
$1.0 : \forall xy. \neg\mathsf{Attends}(x, y) \vee \neg\mathsf{Attends}(x, \text{Course3}) \vee \neg\mathsf{HeldIn}(y, \text{Slot1}) \vee \neg\mathsf{HeldIn}(\text{Course3}, \text{Slot1})$

In general, there can be many choices of generalized axioms. We have to balance the amount of generalization with the number of iterations.

Our generalization algorithm searches over the space of *partial instantiation* of axioms. A partial instantiation of an axiom is one in which some (or all) of the quantified variables of the axiom are ground to particular constants. For instance, $1.0 : \forall xz.F(x, c, z)$ is a partial instantiation of $1.0 : \forall xyz.F(x, y, z)$, where $c \in \mathcal{S}(y)$. An abstract description of our algorithm is shown in Figure 3. It is controlled by a threshold value *thresh* that lies between 0 and 1. The algorithm works by searching over the space of all partial instantiations of axioms, looking for one with a support higher than *thresh*. Here, support of an axiom f is defined as the fraction of formulae in $\mathcal{C}$ that entail f, divided by its total number of instantiations. (The division prevents over generalization.) If one such instantiation is found, we add it to $\mathcal{G}$ and remove the covered axioms from $\mathcal{C}$.

Soft-Cegar uses an efficient implementation of this algorithm. It uses a *thresh* value obtained by training the algorithm on small datasets.

4 Evaluation

In this section, we describe our implementation of Soft-Cegar and compare it with two state-of-the-art relational learning engines Alchemy [12] and Tuffy [16] on four real world applications. All experiments were performed on a 2.66 GHz Intel Xeon quad core processor system with 16 GB RAM running Microsoft Windows Server 2008. Soft-Cegar is implemented in F# and uses the PostgreSQL 8.4 RDBMS engine for storing and manipulating relations. The implementation of Soft-Cegar uses Tuffy as the underlying relational MAP solver (implementation of $\mathsf{Solve}_{\mathsf{map}}$ subroutine in Figure 2). We consider four real-world applications for our evaluation.

Advisor Recommendation (AR). The MLN for this application specifies a recommendation system for starting graduate students that helps them find good PhD advisors. This recommendation is made based on the interests of students (expressed using the Likes(Person, Paper) relation) and weighted formulae such as "prefer an advisor who has graduated a student in an area of interest". The axioms consists of rules such as "the advisor should have had at least one student who is the first author of a paper", in addition to axioms that specify the theory of equality used to discover equivalence classes over papers. The dataset (AIDB) for this application was created from the AI genealogy project (`http://aigp.eecs.umich.edu`) for advisor-student relationships and DBLP (`http://dblp.uni-trier.de`) for bibliographic information. This dataset consists of information about 25 researchers and 600 papers.

Entity Resolution (ER) [23]: This is the problem of identifying duplicate entities in a database. We use the Cora dataset [3] (available at `http://alchemy.cs.washington.edu/data/cora`) that consists of 1295 citations and 132 distinct research papers for our experiments. The objective is to find identical authors, venues, titles and citations in the database. The MLN for this task is described in (`http://alchemy.cs.washington.edu/mlns/er`). Since this is essentially a classification problem, most of the axioms in the MLN are those specifying an equivalence relation.

Information Extraction (IE) [18]: The task for this application is to extract database records from text or semi-structured sources. In our experiment, we use the Cora dataset and the MLN (available at `http://alchemy.cs.washington.edu/mlns/ie`) specifies extraction of author, title and venue fields from this bibliographic dataset. The axioms in this MLN specify the theory of uninterpreted functions [4].

Relational Classification (RC) [16]: In this application, papers in the dataset are classified based on categories of papers published by same authors. The axioms in this MLN specify the theory of equality and uninterpreted functions [4].

The statistics for these four applications (MLNs) and their corresponding datasets is shown in Table 1.

Experiments. Table 2 summarizes the results obtained by running Soft-Cegar, Tuffy and Alchemy over the four applications. The runtime is in minutes:seconds and the cost of a world ω is proportional to negative log of its weight (minimizing this quantity is equivalent to finding the MAP solution).

The rows for Soft-Cegar, Tuffy and Alchemy in Table 2 report the runtimes and solution costs obtained by running these tools over the four applications. The "*" entries in the table correspond to an out-of-memory exception result. For instance, Tuffy runs out of memory on the AR application (after 4 hours), while Alchemy runs out of memory on the IE and RC applications (after 18 hours). On the other hand, Soft-Cegar completes and quickly produces

	AR	ER	IE	RC
#relations	14	14	19	5
#formula	24	3.8K	1.1K	32
#axioms	6	7	3	2
#atoms	88K	20K	81K	9860
#evidence-atoms	65K	676	613K	430K
#query-atoms	188	400	400	400

Table 1. Application MLN and dataset statistics.

Method	Iterations	Time	Solution Cost
Advisor Recommendation			
Soft-Cegar	18	06:44	3669.50
Tuffy	1	-	*
Alchemy	-	-	*
Entity Resolution			
Soft-Cegar	8	13:06	28112.24
Tuffy	1	15:13	34416.97
Alchemy	1	16:17	5287838.62
Information Extraction			
Soft-Cegar	3	17:46	109.40
Tuffy	1	55:49	3944.29
Alchemy	-	-	*
Relational Classification			
Soft-Cegar	2	05:00	870.37
Tuffy	1	05:42	874.63
Alchemy	-	-	*

Table 2. Empirical evaluation of Soft-Cegar.

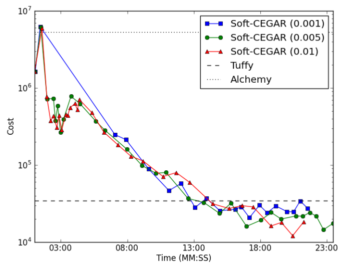

Fig. 4. Cost of solution vs. time for ER on the Cora dataset.

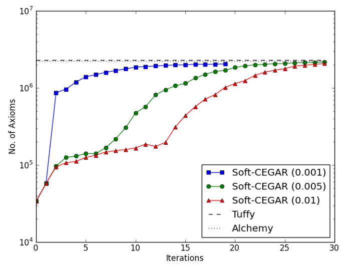

Fig. 5. Number of axioms vs. number of iterations for ER on the Cora dataset.

a result on all applications. On the applications where Tuffy and Alchemy terminate and produce a result, it is important to also note that Soft-Cegar produces solutions which are superior (lower cost) in terms of quality. This empirically supports our intuition that reducing the axiom burden on a relational solver via CEGAR can lead to improvements in efficiency as well as precision.

Figure 4 plots solution cost vs. time for Soft-Cegar, Tuffy and Alchemy on the ER application over the Cora dataset. We have three curves for Soft-Cegar, one for each $thresh = 0.001$, $thresh = 0.005$ and $thresh = 0.01$, where $thresh$ is a parameter that controls the degree of generalization. A lower value of $thresh$ corresponds to more generalization and therefore better acceleration of the CEGAR loop. It can be seen for all curves that the solution cost decreases with time and that Soft-Cegar performs much better than both Tuffy and Alchemy. It can also be seen from Figure 5 that the rate of addition of axioms decreases with increase in values of $thresh$. Note that that Soft-Cegar results in lower cost solutions, validating our intuition that reducing the burden on the relational MAP solver via lazy addition of axioms can also improve the quality of the results. As expected, Tuffy performs much better than Alchemy for this application and this is consistent with the results reported in [16].

For the IE application, Soft-Cegar completes in 3 iterations and this can be attributed to effective generalization which also results in a more precise solution than Tuffy. For the RC application, Soft-Cegar completes in two iterations and this is due to a small number of axioms in this application.

5 Related Work

Several optimizations for scaling statistical relational learning tools have been proposed recently. *Lifted inference* [5, 15, 17] exploits the structure of graphical models to efficiently perform message passing and compute marginal probabilities. In particular, subsets of components that will send and receive identical messages during belief propagation are identified and grouped together for efficiency. *Lazy inference* [19] exploits the observation that most variables in inferred worlds typically have default values (a value that is much more frequent than the others). Thus, we can save both time and memory by assuming that almost all of variables have default values, and gradually refine the variables whose values need to be changed to get an optimum score. Lazy inference generalizes earlier work on Lazy-WalkSAT [24]. *Lazy grounding* approaches have been devised to deal with the size of domains, which are typically very large, by lazily adding constants to be considered from each domain, on demand. Coarse-to-fine inference [10] groups constants from domains into *types* and refines the types progressively to get better scores for the inferred solution. Cutting plane inference [21] starts with a *partial grounding* (which considers only a subset of constants from each domain) with a *partial score*, and iteratively ascertains which constants to add to the grounding so as to improve the score of the inferred solution. The Tuffy [16] tool uses relational database techniques to efficiently perform grounding and avoid constructing groundings that do not matter for calculating the score of the final inferred world.

Our technique for handling axioms lazily is complementary and orthogonal to all these existing optimizations. Tools for relational learning such as Alchemy [12] and Tuffy [16] are publicly available with benchmark suites. Our work was inspired by trying out these tools on these benchmark suites, and observing that CEGAR techniques can significantly improve the performance of these tools. We have implemented our CEGAR based relational learning algorithm in a tool called Soft-Cegar using Tuffy as the underlying relational learner. Tuffy is the most recent optimized relational learning solver, and it incorporates all the optimizations mentioned above from the relational learning community. Our experimental results show that Soft-Cegar outperforms Tuffy, giving empirical evidence that lazily instantiating axioms with CEGAR gives substantial gains over and above all the above optimizations. In addition, our work enables enriching the language of relational learning tools with SMT theory reasoning. Though all our current examples use only the EUF theory, our algorithm works with other theories such as linear arithmetic, which are not currently available in relational learners.

6 Conclusion

Inspired by the wide use of CEGAR (Counterexample Guided Abstraction Refinement) in program verification, we proposed a technique to lazily add axioms in the context of statistical relational inference. We proved that the technique is guaranteed to yield optimal answers assuming that the underlying relational learner is optimal (Theorem 1). In practice, even though existing statistical relational learners use heuristics, and are far from being optimal, we empirically validated that our lazy addition of axioms greatly improves both their runtime and the quality of their solution. The current implementation of our algorithm uses an existing relational learner as a blackbox, and is able to make use of all the orthogonal and complementary optimizations that have been already developed and implemented in existing relational learning tools.

We see several avenues for future work. Our CEGAR framework can be used to add expressiveness to the formulae used in relational learning. In particular, it is possible to integrate theories supported by SMT solvers such as linear arithmetic and provide a more expressive language of formulas for MLNs. This would enable inference to infer constants in the solution that are not present in the evidence or in the domains. While the optimality provided by Theorem 1 is easy to prove for finite domains with such an extension, more work is needed to guarantee optimality and handle infinite domains. Another direction worth exploring is the problem of learning axioms from data using logical reasoning algorithms, inspired by recent approaches proposed to infer the structure of graphical models from data [11, 14].

References

1. T. Ball and S. K. Rajamani. The SLAM project: Debugging system software via static analysis. In *Principles of Programming Languages (POPL 2002)*, pages 1–3, 2002.
2. D. Beyer, T. A. Henzinger, R. Jhala, and R. Majumdar. "The Software Model Checker BLAST". *STTT: International Journal on Software Tools for Technology Transfer*, 9(5-6):505–525, 2007.
3. M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Knowledge Discovery and Data Mining (KDD 2003)*, pages 39–48, 2003.
4. A. R. Bradley and Z. Manna. *The Calculus of Computation: Decision Procedures with Applications to Verification.* Springer-Verlag, 2007.
5. R. Braz, E. Amir, and D. Roth. Lifted first-order probabilistic inference. In *International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 1319–1325, 2005.
6. E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. In *Computer Aided Verification (CAV 2000)*, pages 154–169, 2000.

7. L. de Moura and N. Bjorner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008)*, pages 337–340, 2008.
8. C. Flanagan, R. Joshi, X. Ou, and J. B. Saxe. Theorem proving using lazy proof explication. In *Computer Aided Verification (CAV 2003)*, pages 355–367, 2003.
9. H. Kautz, B. Selman, and Y. Jiang. A general stochastic approach to solving problems with hard and soft constraints. In D. Gu, J. Du, and P. Pardalos, editors, *The Satisfiability Problem: Theory and Applications*, pages 573–586. AMS, 1997.
10. C. Kiddon and P. Domingos. Coarse-to-fine inference and learning for first-order probabilistic models. In *National Conference on Artificial Intelligence (AAAI 2011)*, 2011.
11. S. Kok and P. Domingos. Learning the structure of Markov logic networks. In *International Conference on Machine Learning (ICML 2005)*, pages 441–448, 2005.
12. S. Kok, M. Sumner, M. Richardson, P. Singla, H. Poon, D. Lowd, and P. Domingos. The Alchemy system for statistical relational AI. Technical report, University of Washington, 2007. http://alchemy.cs.washington.edu.
13. K. L. McMillan. Interpolation and SAT-based model checking. In *Computer Aided Verification (CAV 2003)*, pages 1–13, 2003.
14. L. Mihalkova and R. Mooney. Bottom-up learning of Markov logic network structure. In *International Conference on Machine Learning (ICML 2007)*, pages 625–632, 2007.
15. B. Milch, L. S. Zettlemoyer, K. Kersting, M. Haimes, and L. P. Kaelbling. Lifted probabilistic inference with counting formulas. In *National Conference on Artificial Intelligence (AAAI 2008)*, pages 1062–1068, 2008.
16. F. Niu, C. Re, A. Doan, and J. Shavlik. Tuffy: Scaling up statistical inference in Markov logic networks using an RDBMS. In *International Conference on Very Large Data Bases (VLDB 2011)*, 2011.
17. D. Poole. First-order probabilistic inference. In *International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 985–991, 2003.
18. H. Poon and P. Domingos. Joint inference in information extraction. In *National Conference on Artificial Intelligence (AAAI 2007)*, pages 913–918, 2007.
19. H. Poon, P. Domingos, and M. Sumner. A general method for reducing the complexity of relational inference and its application to mcmc. In *National Conference on Artificial Intelligence (AAAI 2008)*, 2008.
20. M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 62:107–136, 2006.
21. S. Riedel. Cutting plane map inference for Markov logic. In *Statistical Relational Learning (SRL 2009)*, 2009.
22. A. Silberschatz, H. Korth, and S. Sudarshan. *Database Systems Concepts*. McGraw-Hill, Inc., 5 edition, 2006.
23. P. Singla and P. Domingos. Entity resolution with Markov logic. In *International Conference on Data Mining (ICDM 2006)*, pages 572–582, 2006.
24. P. Singla and P. Domingos. Memory-efficient inference in relational domains. In *National Conference on Artificial Intelligence (AAAI 2006)*, pages 488–493, 2006.