ECAI 2020
G.D. Giacomo et al. (Eds.)
© 2020 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA200259

STRiKE: Rule-Driven Relational Learning Using Stratified k-Entailment

Martin Svatoš[1] and **Steven Schockaert**[2] and **Jesse Davis**[3] and **Ondřej Kuželka**[4]

Abstract. Relational learning for knowledge base completion has been receiving considerable attention. Intuitively, rule-based strategies are clearly appealing, given their transparency and their ability to capture complex relational dependencies. In practice, however, pure rule-based strategies are currently not competitive with state-of-the-art methods, which is a reflection of the fact that (i) learning high-quality rules is challenging, and (ii) classical entailment is too brittle to cope with the noisy nature of the learned rules and the given knowledge base. In this paper, we introduce STRiKE, a new approach for relational learning in knowledge bases which addresses these concerns. Our contribution is three-fold. First, we introduce a new method for learning stratified rule bases from relational data. Second, to use these rules in a noise-tolerant way, we propose a strategy which extends k-entailment, a recently introduced cautious entailment relation, to stratified rule bases. Finally, we introduce an efficient algorithm for reasoning based on k-entailment.

1 Introduction

One of the key aims of Statistical Relational Learning (SRL) is to develop methods for predicting information that can be plausibly inferred, but is missing from a given set of relational facts K. Our focus in this paper is on rule-based SRL strategies, i.e. our aim is to learn a set of rules Φ which capture regularities that can be observed in K. This set of rules can then be used to infer plausible facts which are missing from K, or even to predict facts which plausibly follow from a new set of relational facts E involving the same predicates as those in K. Using rules is appealing because they allow us to encode learned relational dependencies in a natural and expressive way. However, the main challenge with rule-based approaches is that classical logic is only suitable for encoding hard constraints, whereas most of the learned rules will capture dependencies which are typically but not universally true. Given such a set of learned default rules Φ, the set $E \cup \Phi$ will often be inconsistent, in which case classical entailment cannot be used to derive facts which plausibly follow from E. Classical entailment is similarly unsuitable in settings where the given evidence set E may itself be noisy.

In Markov Logic Networks (MLNs) [27] these limitations of classical logic are avoided by using weighted rules only for defining the potential functions of a Markov network. While this avoids the aforementioned problems with classical entailment, this solution makes the overall approach significantly less transparent, as the rules of an MLN interact in non-trivial and sometimes counter-intuitive ways. One possible alternative, which is used in the AMIE+ system [10], is to only predict facts which can be obtained from a given set of facts E by the single application of a rule from Φ. In this way, errors among the predicted facts are not propagated, and problems with inconsistency are thus avoided completely. However, this approach is clearly very cautious, which can be especially problematic in cases where the evidence in E is sparse. Another alternative, advocated in [15] and [30], relies on using a stratification of the rule base, in the spirit of possibilistic logic [9]. In cases where $E \cup \Phi$ is inconsistent, the least reliable rules are repeatedly removed until consistency is restored. Predictions can then be made by computing the deductive closure with the remaining rules. One of the main conclusions from [15] is that this strategy can be effective in cases where the evidence set E is small, outperforming strategies based on MLNs in such cases. Unfortunately, for larger evidence sets the number of rules that has to be removed often becomes prohibitively large. Consequently, too few meaningful predictions can be made.

The main limitations of these possibilistic logic inspired methods are due to the use of classical entailment, which is intuitively not sufficiently cautious to deal with the default nature of learned rules. The use of stratification can only partially address this problem. On the other hand, the main limitation of AMIE+ is that it eschews entailment completely, which intuitively makes it too cautious. In this paper, we propose a middle ground between these two alternatives. Following [15], we use a stratified set of rules, but we replace classical entailment by k-entailment, a cautious form of logical entailment which limits the extent to which erroneous inferences can propagate. In particular, a formula can be k-entailed from $E \cup \Phi$ if it can be classically entailed from a subset of $E \cup \Phi$ involving at most k constants. We call this approach STRiKE, which stands for STRatified K-Entailment. While various other forms of cautious entailment could be considered, a key advantage of k-entailment is that it admits PAC-style guarantees [13, 35, 18] on the number of incorrect conclusions it infers [16]. While k-entailment has already been studied from a theoretical point of view, no algorithms exist for reasoning based on k-entailment. Moreover, the effectiveness of k-entailment on real datasets has not yet been studied.

The contributions of this paper are as follows. First, we propose a new algorithm for learning stratified rule bases which is suitable for the considered setting. Our method is orders-of-magnitude faster than the method from [15], thanks to the fact that our weight learning strategy avoids the need for model counting. Compared to the rule-learning strategy from AMIE+ [10], our approach is able to find more natural theories. This is because our approach learns the stratified rule base jointly, that is, it considers the impact of adding each candidate on the rule base as a whole. In contrast, AMIE+ evaluates each rule in isolation and hence does not consider how the rule un-

[1] CTU in Prague, Czech Republic, email: svatoma1@fel.cvut.cz
[2] Cardiff University, United Kingdom, email: schockaerts1@cardiff.ac.uk
[3] KU Leuven, Belgium, email: jesse.davis@cs.kuleuven.be
[4] CTU in Prague, Czech Republic, email: kuzelon2@fel.cvut.cz

der construction interacts with previously learned rules. Furthermore, our approach shares the advantage of [15] that standard SAT solvers can be used to prune the learned theories, which can in some cases lead to particularly compact and natural representations. Second, we introduce the first algorithm for reasoning with k-entailment, relying on ideas from relational databases to enable efficient inference, an idea which we borrow from Markov Logic Network (MLN) reasoners [21, 28]. Finally, we experimentally demonstrate the effectiveness of the proposed stratified k-entailment approach.

2 Background

We consider a standard function-free first-order language defined by a set of constants $\mathcal{C}$, a set of variables $\mathcal{V}$ and for each $k \in \mathbb{N}$ a set $\mathcal{R}_k$ of k-ary predicates. To avoid confusion, variables start with lowercase letters and constants start with uppercase letters. An atom is of the form $r(a_1, ..., a_k)$ with $a_1, ..., a_k \in \mathcal{C} \cup \mathcal{V}$ and $r \in \mathcal{R}_k$. A literal is an atom or its negation. A clause is a universaly quantified disjunction over a finite set of literals. We assume that the variables in a clause are all universally quantified. For a (set of) clause(s) Θ, $\textbf{\textit{Const}}(\Theta)$ denotes the set of constants appearing in Θ. For a set of constants $\mathcal{C}_0 \subseteq \mathcal{C}$, $\Theta[\mathcal{C}_0]$ denotes the set of clauses obtained from Θ by removing all clauses that contain a constant $C \notin \mathcal{C}_0$. For instance, when $\Theta = \{\mathit{Friends}(\mathit{Alice}, \mathit{Bob}), \mathit{Smokes}(\mathit{Alice})\}$ then $\Theta[\{\mathit{Alice}\}] = \{\mathit{Smokes}(\mathit{Alice})\}$. A clause in which none of the literals contains any variables is called *ground*. The grounding of a clause α w.r.t. a set of clauses Θ is the set $G_\Theta(\alpha) = \{\alpha\theta_1, ..., \alpha\theta_m\}$ of all ground clauses that can be obtained by substituting the variables occurring in α by constants from $\textbf{\textit{Const}}(\Theta)$. When Θ is clear from the context, we also write $G_\Theta(\alpha)$ as $G(\alpha)$.

We will mostly focus on sets of *facts*, *definite rules* and *constraints* instead of full first-order logic theories. These terms are understood as in the *logic programming* literature [17]. In particular, a *fact* is a ground positive literal. A *definite rule* is a clause which has exactly one positive literal; note that facts are special cases of definite rules. A *constraint* is a clause that has no positive literals. To improve readability, we will usually write a definite rule $\forall x_1, ..., x_m : h \vee \neg b_1 \vee \cdots \vee \neg b_n$ as $\forall x_1, ..., x_m : b_1 \wedge \cdots \wedge b_n \rightarrow h$. As usual, we call $b_1 \wedge ... \wedge b_n$ the body of the rule and h the head. Unless specified otherwise, we will restrict ourselves to working with range-restricted rules, which are rules that satisfy $\mathbf{Vars}(h) \subseteq \mathbf{Vars}(b_1 \wedge ... \wedge b_n)$.

A fact β is classically entailed by a set of clauses $\mathrm{A} = \{\alpha_1, ..., \alpha_n\}$, written $\mathrm{A} \models \beta$, iff $G_\mathrm{A}(\alpha_1) \cup ... \cup G_\mathrm{A}(\alpha_n) \models \beta$, with $\models$ in the latter case denoting the standard entailment relation from propositional logic. When restricting ourselves to facts and definite rules we can use forward chaining to compute the set of facts that are entailed by a given set of clauses. As a result, classical entailment is tractable in this case. It also means that the inference process can be easily explained to users who are not familiar with formal logic.

Example 1. *Forward chaining iteratively expands the given set of facts by applying (groundings of) rules whose body is satisfied by the facts that have already been derived. More precisely, if all the literals in a ground rule's body are included in the set of facts, then its head can be added to the set of facts. This process is repeated until no further facts can be derived. To illustrate this, consider the following set of rules:*

$$\forall x, y, z : \mathit{BornIn}(x, y) \wedge \mathit{PartOf}(y, z) \rightarrow \mathit{BornIn}(x, z)$$
$$\forall x, y : \mathit{BornIn}(x, y) \wedge \mathit{Country}(y) \rightarrow \mathit{Nationality}(x, y)$$

and the following set of facts:

$$F_1 = \{\mathit{BornIn}(\mathit{Alice}, \mathit{SdC}), \mathit{Country}(\mathit{Spain}), \mathit{PartOf}(\mathit{SdC}, \mathit{Spain})\}.$$

We can use the first rule to derive $\mathit{BornIn}(\mathit{Alice}, \mathit{Spain})$*. Together with the fact* $\mathit{Country}(\mathit{Spain})$ *from* F_1*, we can now apply the second rule to derive* $\mathit{Nationality}(\mathit{Alice}, \mathit{Spain})$*. At this point, no further facts can be derived.*

3 Bounded Reasoning Using k-Entailment

We now describe the notion of k-entailment that was introduced in [16] for reasoning in relational domains, but we specialize it to our rule-based setting. Intuitively, a fact can be k-entailed from a given rule base if it has a derivation involving at most k constants. This captures the intuition that, in most domains, the knowledge about one specific constant typically only directly affects what we can derive about a small set of closely related constants. For instance, when modelling a social network, using k-entailment means that the knowledge we have about a given user only directly affects what we derive about the users in their neighborhood. By using 2-entailment, we thus restrict the impact of any errors in our knowledge about some user to his or her direct friends. Similarly, using 3-entailment would mean that our knowledge about the friends of a user's friends might also be affected, but not the wider network. In contrast, when using classical logic, a single error might in principle affect the entire network. Formally, k-entailment is defined as follows.

Definition 1 (k-entailment)**.** *Let k be a non-negative integer, E a set of facts, Φ a set of rules and Γ a set of constraints. We say that a fact α is k-entailed by $E \cup \Phi \cup \Gamma$, denoted $E \cup \Phi \cup \Gamma \models_k \alpha$, if there is some $C \subseteq \textbf{\textit{Const}}(E \cup \Phi \cup \Gamma)$ such that: (i) $|C| \leq k$, (ii) $(E \cup \Phi \cup \Gamma)[\mathcal{C}]$ is consistent, and $(E \cup \Phi)[\mathcal{C}] \models \alpha$.*

Note that, apart from limiting the impact that a given piece of knowledge can have, in the presence of constraints, k-entailment also blocks all inferences involving sets of constants $\mathcal{C}$ that make $(E \cup \Phi \cup \Gamma)[\mathcal{C}]$ inconsistent. This is illustrated in the next example.

Example 2. *Let us consider the following:*

$$\begin{aligned} \Phi &= \{\forall x : \mathit{Giraffe}(x) \rightarrow \mathit{Animal}(x), \\ &\quad\ \forall x, y : \mathit{Friends}(x, y), \rightarrow \mathit{Friends}(y, x), \\ &\quad\ \forall x, y : \mathit{Friends}(x, y) \rightarrow \mathit{Human}(x)\} \\ \Gamma &= \{\forall x : \neg \mathit{Human}(x) \vee \neg \mathit{Animal}(x)\} \\ E &= \{\mathit{Giraffe}(\mathit{Liz}), \mathit{Friends}(\mathit{Ann}, \mathit{Liz})\}. \end{aligned}$$

Then $\mathit{Animal}(\mathit{Liz})$ is the only fact which is 2-entailed by $E \cup \Phi \cup \Gamma$, besides those in E. In particular, note that we can infer neither $\mathit{Human}(\mathit{Liz})$ nor $\mathit{Human}(\mathit{Ann})$ because $(E \cup \Phi \cup \Gamma)[\{\mathit{Ann}, \mathit{Liz}\}]$ is not logically consistent. Intuitively, the constraint in Γ and rules in Φ assert that animals do not have friends and that one cannot be an animal and a human at the same time. This means that, assuming the rules were perfect, $\mathit{Giraffe}(\mathit{Liz})$ and $\mathit{Friends}(\mathit{Ann}, \mathit{Liz})$ cannot both be correct. Since we do not want the potential errors to spread through our knowledge base, by using 2-entailment we avoid inferences that involve Ann and Liz at the same time.

In [16] we showed that k-entailment allows deriving PAC-type guarantees for reasoning in relational domains. The main idea is as follows. Suppose we have a set of rules and constraints $\Phi \cup \Gamma$. We randomly pick a subset $\mathcal{S}$ of k objects (first-order logic constants) from the domain. We define "accuracy" as the probability that all rules and constraints from $\Phi \cup \Gamma$ are satisfied in the fragment induced by $\mathcal{S}$. We can then use the accuracy defined in this way to bound the number of atoms that k-entailment predicts as true but which should actually be false. In particular, we have for the set of incorrectly predicted atoms $\mathcal{F}$ that $|\mathcal{F}| \leq (1 - Acc)\,|\mathcal{C}|^k k^a$, where $\mathcal{C}$ is the domain, Acc is the accuracy of $\Phi \cup \Gamma$ and a is the maximum arity of the atoms. We also showed that such a bound would not work if we used classical reasoning due to its brittleness. Similarly, such guarantees cannot be provided for most of the approaches that have been developed for inconsistency-tolerant reasoning, e.g. those based on maximal consistent subsets [3, 11], since such approaches usually coincide with classical entailment in the absence of inconsistencies, and clearly errors can still be propagated even in consistent knowledge bases.

The significance of the insights from [16] for this paper is that they justify why our rule learner is able to evaluate candidate rules based on how accurate they are on small fragments of the training data (assuming that this training data was sampled from a reasonable distribution). Our rule learning approach in Section 6 will take advantage of this insight.

4 Inference Algorithm

Naively following the definition of k-entailment would require enumerating all subsets of up to k constants in the given domain Δ and determining which literals can be entailed from each of them. This section shows how to implement k-entailment in a much more efficient way. In particular, Algorithm 1 presents an approach which is based on a forward-chaining procedure with additional bookkeeping. Specifically, the algorithm solves the following problem:

Given: *A set of rules* $\Phi = \{\alpha_1, \ldots, \alpha_l\}$*, constraints* $\Gamma = \{\beta_1, \ldots, \beta_{l'}\}$ *and facts* $E = \{a_1, a_2, \ldots, a_n\}$.

Compute: *The set of positive literals* k*-entailed by* $\Phi \cup \Gamma \cup E$.

We assume w.l.o.g. that Φ does not contain any facts, noting that including a fact α in Φ is equivalent to including α in E.

4.1 Description of the Algorithm

The algorithm begins by initializing a hash table *Support*, which associates literals (keys) to sets of subsets of constants (values). When the algorithm terminates, *Support*$[a]$ will contain all minimal sets of constants C, up to size k, for which $E[C] \cup \Phi \cup \Gamma \models a$. The set of k-entailed literals will thus correspond to the key set of *Support*, which we denote by ***Keys***(*Support*). To find these sets C, the algorithm alternatingly adds new candidate subsets of constants and then filters those which turn out to be inconsistent. To detect inconsistencies efficiently, the algorithm maintains the set *Incons*. This set is initially empty. Throughout the execution of the algorithm, it is used to store all encountered subsets I of $\mathcal{C}$ where $E[I] \cup \Phi \cup \Gamma$ is found to be logically inconsistent.

The first step of the main loop is to solve the following subproblem: given a set of ground positive literals E and a rule $\alpha = (b_1 \wedge \cdots \wedge b_l \rightarrow a)$ or a constraint $\beta = \neg(b_1 \wedge \cdots \wedge b_l)$, find the set of all groundings ϑ such that $(b_1 \wedge \cdots \wedge b_l)\vartheta \subseteq E$. We will denote the corresponding set of ground rules and constraints by ***Active***(α, E). In other word, this set contains those ground rules and constraints whose body is satisfied by the literals that we have derived so far. It can be efficiently computed using a conjunctive query in a relational database, similar to how such relational database engines are used in Markov logic inference [21, 28].

In Step 2, the algorithm removes all active rules α for which there there is some $I \in$ *Incons* such that $I \subseteq$ ***Const***(α). The rules which are removed in this step correspond to those whose body can only be satisfied by starting from a fragment of E that is inconsistent with $\Phi \cup \Gamma$ (or more precisely, those which have previously been found to be inconsistent).

In Step 3, the algorithm iterates over the remaining active rules. For each such rule $(b_1 \wedge \cdots \wedge b_m) \rightarrow h \in \mathcal{A}$ it combines the support sets of the ground literals $b_1, \ldots, b_m$ and stores them in a set $\mathcal{U}$. In particular, if we have $S_1 \in$ *Support*$[b_1]$, this intuitively means that we can derive b_1 using a fragment of E that only involves the constants from S_1, and similar for $b_2, \ldots, b_m$. Hence, thanks to the given rule, this means that we can also derive its head h using a fragment that contains all constants from $S = S_1 \cup \cdots \cup S_m$. Of these sets S, we only keep those which have at most k elements and which are minimal (w.r.t. set inclusion). Next, the procedure ***FindInconsistent*** aims to detect subsets of constants C that occur in *Support*$[h']$, for some literal h', such that $E[C] \cup \Phi \cup \Gamma$ is inconsistent. To do this efficiently, we take advantage of the fact that each iteration of the main loop corresponds to an iteration of a modified forward chaining procedure. Rather than checking whether $E[C] \cup \Phi \cup \Gamma$ is inconsistent, we thus check whether such an inconsistency has been derived so far. This is done in two steps (not shown in the pseudo-code). First, we determine all the sets of constants C whose consistency needs to be (re-)checked. This is the case for all supersets of the elements from $\mathcal{U}$ (including the elements from $\mathcal{U}$ themselves). For the second step, let us write *Support*$^{-1}[\mathcal{A}]$ for the set of all literals h' for which *Support*$[h']$ contains a subset of $\mathcal{A}$. Note that *Support*$^{-1}[\mathcal{A}]$ intuitively corresponds to the set of literals for which our forward chaining procedure has already established that they can be derived from $E[\mathcal{A}] \cup \Phi \cup \Gamma$. For every superset $\mathcal{A}$ of an element from $\mathcal{U}$, the procedure checks if *Support*$^{-1}[\mathcal{A}]$ is consistent with the constraints in Γ; it returns all sets $\mathcal{A}$ for which this is not the case. Finally, these sets of constants are removed from *Support*. If the set of support sets for some literal h' in ***Keys***(*Support*) then becomes empty, this literal is removed from the key set of the hash table. Finally, all support sets found as inconsistent are added to the set *Incons*.

In Steps 4 and 5, the algorithm first updates the current state of the evidence E to contain exactly the facts which occur as keys in the hash table *Support* and then it either goes back to Step 1 if *Support* has been modified in the current iteration, or it finishes and returns E.

We give a sketch of correctness in the appendix. Next we describe an illustrating example.

4.2 An Illustration

Here we exemplify one run of the algorithm. Let Φ, Γ and E be as in Example 2, which talks about the giraffe *Liz*. Running the initialization procedure will result in *Support* $= \{$*Giraffe*(*Liz*) : $\{$*Liz*$\}$, *Friends*(*Ann*, *Liz*) : $\{$*Ann*, *Liz*$\}\}$. Now, in the first step, we find the set $\mathcal{A}$, which is the set of ground rules whose bodies are true in the current E (we call these rules "active"):

$$\begin{aligned}\mathcal{A} = \{&\mathit{Giraffe}(\mathit{Liz}) \Rightarrow \mathit{Animal}(\mathit{Liz}),\\ &\mathit{Friends}(\mathit{Ann},\mathit{Liz}) \Rightarrow \mathit{Friends}(\mathit{Liz},\mathit{Ann})\},\\ &\mathit{Friends}(\mathit{Ann},\mathit{Liz}) \Rightarrow \mathit{Human}(\mathit{Ann})\}.\end{aligned}$$

After the loop on line 3 is executed, the state of the hash table becomes *Support* $= \{$*Giraffe*(*Liz*) : $\{$*Liz*$\}$, *Friends*(*Ann*, *Liz*) : $\{$*Ann*, *Liz*$\}$, *Animal*(*Liz*) : $\{$*Liz*$\}$, *Friends*(*Liz*, *Ann*) :

```
Initialization:
1. For $\forall a \in E$, set $Support[a] := \{\textbf{Const}(a)\}$
2. Set $Incons := \emptyset$

Main Loop:
1. Let $\mathcal{A} := \cup_{\alpha \in \Phi} \textbf{Active}(\alpha, E)$.
2. Remove from $\mathcal{A}$ all ground rules $\alpha$ for which there is $\mathcal{I} \in Incons$
   such that $\mathcal{I} \subseteq \textbf{Const}(\alpha)$.
3. For every ground rule $b_1 \wedge \cdots \wedge b_m \rightarrow h \in \mathcal{A}$ do:
   (a) $\mathcal{S}_{Cart} := Support[b_1] \times \cdots \times Support[b_m]$
   (b) $\mathcal{U} := \{S_1 \cup \cdots \cup S_m | (S_1, \ldots, S_m) \in \mathcal{S}_{Cart}\}$
   (c) $Support[h] := Support[h] \cup \{S \in \mathcal{U} | |S| \leq k\}$
   (d) /* Remove non-minimal supports: */
       $Support[h] := \{\mathcal{A} \in Support[h] | \nexists \mathcal{A}' \in Support[h] : \mathcal{A}' \subsetneq \mathcal{A}\}$
   (e) $I := \textbf{FindInconsistent}(h, Support, \Gamma)$
   (f) For all $h'$ in $\textbf{Keys}(Support)$, we set:
       $Support[h'] := Support[h'] \setminus I$
       If this results in $Support[h'] = \emptyset$ then $h'$ is removed from
       $\textbf{Keys}(Support)$.
   (g) $Incons := Incons \cup I$
4. $E := \textbf{Keys}(Support)$
5. If Support was changed in the last iteration, go back to Step 1.
   Else finish and return $E$.
```

Algorithm 1: k-Entailment Inference Algorithm

$\{Ann, Liz\}, Human(Ann) : \{Ann, Liz\}\}$. Next, in the procedure ***FindInconsistent***, none of the support sets will be found inconsistent in this iteration because there is no inconsistency with the constraints Γ. The algorithm sets $E := \textbf{Keys}(Support)$ and continues from line 1.

In the second iteration, the algorithm then adds $Human(Liz) : \{Ann, Liz\}$ to *Support*. Next, in step 3e, the algorithm finds the following support sets to be inconsistent $Incons = \{\{Ann, Liz\}\}$. After the filtering step, the state of the hash table *Support* becomes

$$Support := \{Animal(Liz) : \{Liz\}, Giraffe(Liz) : \{Liz\}\}.$$

In the third (and in this case final) iteration, the state of the hash table *Support* remains the same and the algorithm finishes, returning the set $\{$ *Animal*(Liz), *Giraffe*(Liz), *Friends*(Ann, Liz) $\}$. Note that the fact *Friends*(Ann, Liz) is added from the original evidence set.

5 Stratified k-Entailment

Unlike classical logic, k-entailment is not monotonic. In particular, $E \cup \Phi \cup \Gamma \models_k \alpha$ does not imply that $E \cup E' \cup \Phi \cup \Gamma \models_k \alpha$, for E, E' sets of ground literals, Φ a set of rules and Γ a set of constraints. This is illustrated in the next well-known example from the non-monotonic reasoning literature.

Example 3. *Consider the following rules and constraints:*

$$\Phi = \{\forall x : Bird(x) \rightarrow Flies(x), \forall x : Penguin(x) \rightarrow Bird(x)\}$$

$$\Gamma = \{\forall x : \neg Penguin(x) \vee \neg Flies(x)\}.$$

For $E = \{Bird(Tweety)\}$, we can derive Flies$(Tweety)$ using k-entailment. However, when we additionally know that Tweety is a penguin, i.e. if our evidence is given by $E' = \{Bird(Tweety), Penguin(Tweety)\}$, we can no longer derive Flies$(Tweety)$ using k-entailment because $(\Phi \cup \Gamma \cup E')[\{Tweety\}]$ is inconsistent.

This non-monotonic behavior is in itself not problematic. Indeed, it is standard practice in AI to use non-monotonic reasoning when dealing with rules that may have exceptions, and probabilistic reasoning is also non-monotonic. However, the behavior of k-entailment in the presence of conflicts is arguably too cautious: most frameworks for non-monotonic reasoning would still derive *bird*$(Tweety)$ from *penguin*$(Tweety)$ in the previous example. This can be achieved by taking into account that the rule $\forall x : Penguin(x) \rightarrow Bird(x)$ is more reliable than the other rule, either by inducing an ordering on the set of rules automatically [24, 12] or by relying on an explicitly given ordering of these rules [4].

We will follow the latter strategy to obtain a refinement of k-entailment, which we call stratified k-entailment or STRiKE.

Definition 2 (Stratified k-Entailment)**.** *Let $\Lambda = (\alpha_1, \ldots, \alpha_m)$ be a list of rules and constraints and let E be a set of facts. We say that a fact a is k-entailed at level i from Λ and E if there exists some $j \leq i$ s.t. $\{\alpha_1, \ldots, \alpha_j\} \cup E \models_k a$.*

The general intuition is that the rules and constraints in Λ are ordered based on how confident we are in them (i.e. α_1 is the most confident rule or constraint). The use of stratified k-entailment serves two purposes. First, ordering the rules and constraints permits ordering the predictions made by stratified k-entailment according to how confident we are in them. This brings k-entailment closer to approaches such as AMIE+ and MLNs, which also provide confidence values for the predicted facts. In applications, this allows us to tune the trade-off between precision and recall. Second, by taking the ordering of the rules and constraint into account, we can avoid the situation from Example 3, where a less reliable rule was blocking the conclusion of a more reliable rule. This is illustrated in the next example.

Example 4. *Consider the following list: $\Lambda = (\forall x : Penguin(x) \rightarrow Bird(x), \forall x : \neg Penguin(x) \vee \neg Flies(x), \forall x : Bird(x) \rightarrow Flies(x))$ and let $E = \{Penguin(Tweety)\}$. Then one can check that Bird$(Tweety)$ is k-entailed at level 1 whereas with standard k-entailment we could not derive Bird$(Tweety)$ at all.*

Note that the inference mechanism in the previous example is similar in spirit to the one from possibilistic logic [9]. However, the ordering of formulas in possibilistic logic plays a different role than in stratified k-entailment: it is used in possibilistic logic to avoid entailment becoming trivial in the face of inconsistencies, but in our setting, the use of k-entailment already prevents entailment from becoming trivial. Furthermore, standard possibilistic logic only considers propositional formulas. The set of entailed facts in possibilistic logic is also unordered, when the standard approach to inconsistency handling is used. Specifically, a formula is entailed from a possibilistic logic knowledge base iff it can be clasically entailed from the set of formulas above the so-called inconsistency level, i.e. the first level i such that $\{\alpha_1, ..., \alpha_i\}$ is inconsistent.

6 A Heuristic Rule Learner

We introduce a heuristic algorithm for learning rules that are suitable for reasoning with stratified k-entailment. At a high level, the algorithm performs a beam-search through the space of definite rules, using a refinement operator [20] that adds one literal at a time to the rules in the beam. This is a standard strategy in relational learning and inductive logic programming systems [20, 25]. The algorithm's key novelty is in how it heuristically scores the candidate rules.

For a rule $\alpha = b_1 \wedge \cdots \wedge b_m \rightarrow h$ and a set of facts E, we define $\tau_\Lambda(\alpha) = |H_\alpha \cap E|/|H_\alpha|$ where $H_\alpha = \{h\vartheta | E \models (b_1 \wedge \cdots \wedge b_m)\vartheta, \vartheta \in G_E(\alpha)\}$. In other words, $\tau_\Lambda(\alpha)$ corresponds to the percentage of facts predicted by the rule α that we know to be true, that is, the *precision* of the rule. We then treat a list of rules $\Lambda = (\alpha_1, \ldots, \alpha_l)$, where $\tau_\Lambda(\alpha_i) \geq \tau_\Lambda(\alpha_{i+1})$, as defining a probabilistic model that assigns to any fact a the probability $p(a) = \max\{\tau_\Lambda(\alpha_i) \,|\, 1 \leq i \leq l, a \in H_{\alpha_i}\}$. Thus, this simple model assumes that the probabilities of the facts are independent. However, we recall that this is still just a heuristic for selecting the rules, not the final model for prediction. The advantage of this probabilistic view is that it naturally allows us to learn rules by selecting those that most improve the log-likelihood of the training data, i.e. $\sum_{a \in E} \log p(a) + \sum_{a \in E^C} \log(1 - p(a))$. This strategy contrasts with the standard covering strategy used in inductive logic programming [19], which is brittle and has problems with imbalanced data. Our rule learning strategy exploits the properties of *stratified* k-entailment, especially its ability to predict facts with different levels of certainty. In the classical inductive logic programming setting, one would normally optimize accuracy (or a closely related measure) and would therefore often only end up with high precision rules. In contrast, since stratified k-entailment ranks the predicted facts by confidence, it can also work well with less precise rules (although the usefulness of such lower-confidence predictions clearly depends on the application).

In order to force the rule learning algorithm to discover non-trivial relationships, we give it subsampled data. Consider, for instance, that the training dataset contains facts of the form *Friends*(x, y) and that it is complete. Then it would be difficult to learn any rules beyond $\forall x, y :$ *Friends*$(x, y) \rightarrow$ *Friends*(y, x) as no other rule for predicting friendship would improve the log-likelihood. However, by subsampling the data we can break these symmetries, which means that other rules may be found that improve the log-likelihood. For instance, assume that the fact *Friends*(C, A) is missing from the subsampled dataset, but that the facts *Friends*(A, B), *Friends*(B, C) and *Friends*(A, C) are present. Then the rule learner might add the rule $\forall x, y, z :$ *Friends*$(x, y) \wedge$ *Friends*$(y, z) \rightarrow$ *Friends*(x, z), which might improve log-likelihood because it can be used to predict *Friends*(A, C). In contrast, without subsampling, *Friends*(A, C) can simply be predicted from *Friends*(C, A) using the symmetry rule, hence there would not be any reason for adding the transitivity rule.

Finally note that given a list of rules $\Lambda = (\alpha_1, \ldots, \alpha_m)$, there often exists a probability-assigning function $\tau'_\Lambda(.) \neq \tau_\Lambda(.)$ that will lead to a better log-likelihood score; in fact, an optimal one can be obtained using the geometric programming formulation from [15]. However, using the fixed distribution $\tau_\Lambda(.)$ has several practical advantages. First, it means that all rules and their weights can be understood in isolation from the other rules and their weights: a rule's weight gives a lower bound on the probability of the facts it predicts. Second, it means that adding more rules at some point stops improving the log-likelihood even if there are some not yet used rules, which we think of as a form of heuristic regularization.

7 Experiments

In this section, we experimentally evaluate our proposed approach STRiKE. The goal of our empirical evaluation is to address the following two questions:

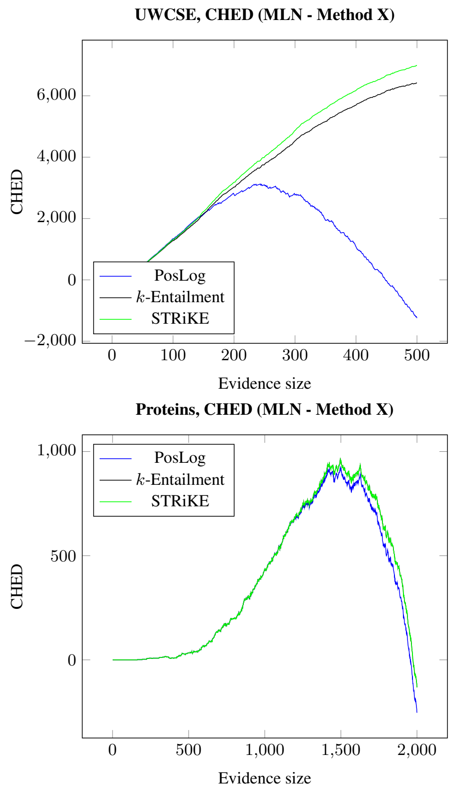


Figure 1: Comparison of the performance for inference given a fixed rule set. The results show the cumulative Hamming error as a function of the size of the evidence set between the predictions made by (1) PosLog, (2) k-entailment, and (3) STRiKE versus MLN-MAP. Increasing slopes mean that a specific inference approach outperforms MLN-MAP.

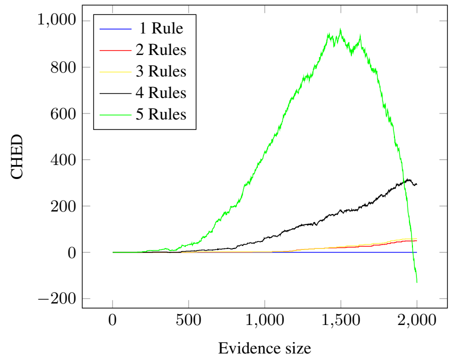
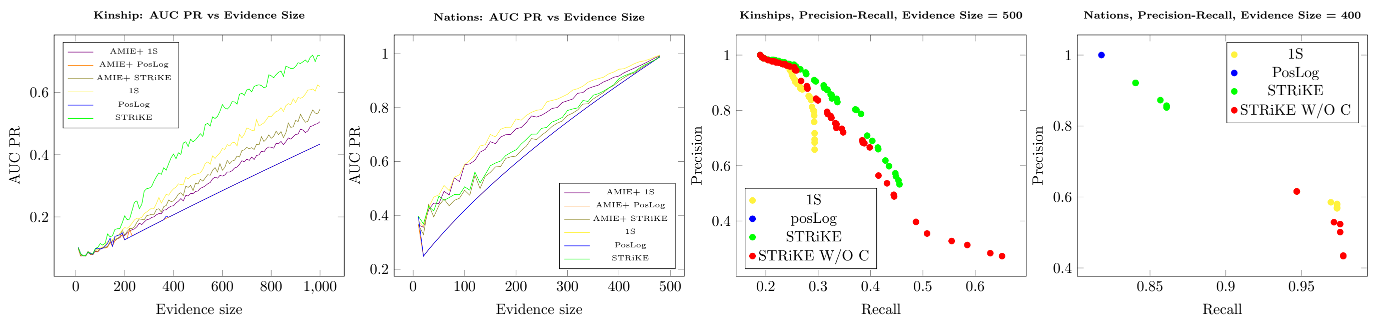


Figure 2: The cummulative difference of Hamming errors of STRiKE vs MLN-MAP on the proteins dataset for rule sets with 1, 2, . . . , 5 best rules. Increasing slopes mean that STRiKE inference outperforms MLN-MAP.

1. How does our proposed STRiKE reasoning method compare to other forms of inference?
2. Does our proposed rule learner learn more accurate theories than existing approaches?

In all the experiments reported in this section, we set the parameter



(a) (b) (c) (d)

Figure 3: The left two panels, 3a and 3b, show the AUC PR as a function of size of the evidence when combining the rules learned by STRiKE or AIME+ with various inference strategies. The right panels, 3c and 3d, show precision-recall curves for two representative evidence set sizes.

of (stratified) k-entailment to $k = 5$.[5]

7.1 Evaluation of STRiKE Inference

To address the first question, we compare the following four approaches to reasoning with a fixed set of relational rules:

MLN-MAP uses RockIt [22] to perform MAP-inference in a Markov logic network learned using the default structure learner from the Alchemy package.

PosLog is the possibilistic logic inference approach from [15].

k**-entailment** is the k-entailment algorithm proposed in Section 4.

STRiKE is our stratified k-entailment proposed in Section 5.

We compare these four approaches on the UWCSE and proteins data set, which were used in [15].[6] We use the same sets of rules, MLNs and constraints as in [15]. We follow the same experimental protocol as [15]. In particular, we randomly divide the constants into two disjoint sets of equal size. The training set consists of atoms containing only the constants from the first set and the test set contains only the constants from the second set. We then predict the set of facts given evidence sets of increasing size using the four inference methods and compute the Hamming error, which measures the size of the symmetric difference of the predicted set of facts and the set of facts in the test set. We then report the cumulative differences between the errors of the models and a baseline, which is MAP-inference in Markov logic networks.

The results are shown in Figure 1. A specific inference method outperforms MAP inference in MLNs for a range of evidence set-sizes if the curve is increasing and performs worse otherwise. Steeper curves signify large differences in performance.

In this case, STRiKE performed best among the considered methods. For UWCSE, it improves MLNs consistently, across the entire range of evidence sizes, in contrast to the possibilistic logic strategy which is only effective for sufficiently small evidence sets. On the proteins dataset, we can see that MLNs perform best for large evidence sets; the difference between STRiKE and possibilistic logic is also smaller on this dataset. The relatively weak performance of this variant, shown in Figure 1, shows that having constraints can be beneficial. On the Proteins dataset, on the other hand, there was no difference between the variants with and without constraints.

To better understand why MLNs outperformed STRiKE for large evidence sets on the Proteins dataset, we perform the following experiment. We select the n most confident rules from the rule set and then report the cumulative Hamming error of predictions made by STRiKE and MLN-MAP when using the simplified rule set. Figure 2 shows the results for this experiment when using only top $n = \{1, \ldots, 5\}$ rules. Having all but the least certain rule already leads to STRiKE outperforming MLNs over the entire range of evidence set sizes.

7.2 Evaluation of the Heuristic Rule Learner

We now compare the predicted performance for the theories learned by our rule learner to AIME+, which is a state-of-the-art relational rule learner [10]. In order to obtain a subsampled dataset for learning definite rules, we uniformly select half of the facts. Constraints are learned exhaustively by searching through all possible size-2 constraints on the training data. After learning a theory, we make predictions using three inference methods: (1) STRiKE, (2) possibilistic logic (*PosLog*), and (3) one-step prediction (*1S*), which returns the predictions made by performing only one iteration of forward chaining. We write *STRiKE* for the version using our rules and *AMIE+ STRiKE* for the version with AMIE+'s rules, and similar for *PosLog* and *1S*.

We followed the same experimental protocol as in Section 7.1, but report the area under the precision-recall curve (AUC PR) because AUC PR is better suited for evaluating weighted predictions. We used the following settings for our rule learner: beam-size was set to 4 and the number of iterations of beam search was set to 5. Both rule learners, AMIE+ and ours, shared the following settings: minimum coverage was set to 1, maximum number of literals in the rules' bodies was set to 3, and the maximal number of variables within a rule was limited to 5. From AMIE+'s rules we selected the five highest-confidence rules per head-predicate because AMIE+ usually returns tens or hundreds of thousands of rules and using all of them would make performing inference on the test data extremely slow. Hence we chose settings that would give us the same number of rules from AMIE+ as from our rule learner. This also means that the improvements seen in experiments for our rule learner are partially due to the heuristic strategy used by our rule learner to select the rules.

The results are shown in Figure 3. Figure 3a and 3b display AUC PR of the methods as a function of evidence size. Figure 3c and 3d display two illustrative PR curves for different evidence sizes on the two datasets; these allow us to gain better insight into the behavior of the methods. Here, we also added STRiKE without any constraints for comparison. As can be seen from these graphs, using constraints

[5] Available from `https://github.com/martinsvat.`
[6] Both available from `http://alchemy.cs.washington.edu/data/`

helps STRiKE to obtain better precision but at the cost of decreasing the recall, which is to be expected.

We observe the following trends. First, rules learned using our heuristic method perform consistently better than rules obtained by AMIE+ (for the same inference method). In fact, in some cases AMIE+ did not manage to find rules beyond length 3, even after several hours, whereas our rule learner had no problems finding longer rules (Table 1). However, it should be noted that the two algorithms are quite different and were designed with different goals in mind. Second, the methods generally work as expected. STRiKE without constraints (STRiKE W/O C) derives more than STRiKE and 1S but its precision is lower than that of STRiKE. Finally, possibilistic logic performs clearly worse, except for very small evidence sets.

Table 1: Runtimes in minutes of rule learners on both datasets given maximal literals in rule body.

domain	Nations		Kinships	
no. literals / method	STRiKE	AMIE+	STRiKE	AMIE+
3	3	50	90	0.1
4	2.5	X > 24h	0.1	417

8 Related Work

Our approach can be seen as an alternative to Markov Logic Networks (MLNs) [27]. MLNs address the brittleness of classical logic by using weighted rules that define a probabilistic graphical model. Among others, this means that MLNs can be used for marginal inference, which our method does not support. However, learning MLNs is challenging and their effectiveness for knowledge base completion tasks is not well-understood. The same holds to a large extent for other statistical relational learning systems (e.g. [8, 1]) as well. Another popular framework for completing knowledge bases consists in learning predictive vector space embeddings of objects and predicates [2, 33, 5]. However, such knowledge graph embedding methods lack transparency and they tend to perform poorly when some sets of objects are only sparsely represented in K [34]. There are also methods that combine rule-based reasoning and vector space embeddings, e.g. [31, 29] but, to a large extent, these methods also share some of the limitations of the other knowledge graph embedding methods.

Our work is also related to approaches for learning relational rules from data. The most closely related work falls in the area of structure learning for statistical relational learning (e.g., [14, 7]). These approaches typically evaluate a candidate rule's usefulness in the context of the current model. They also often employ a beam search. However, for formalisms like Markov logic, unless one considers a restricted model class, it is intractable to select rules that maximize the log likelihood of the data. For this reason, an approximate measure, such as pseudo likelihood, is often used instead (e.g., [14]). Our approach to rule learning is quite different from traditional approaches to inductive logic programming (e.g., [32, 26]), which typically employ a cover-removal style approach that scores each rule independently by looking at, for example, the difference in the number of positive and negative examples a rule covers. Similarly, more recent approaches to mining relational rules from KBs (e.g., [10, 37]) continue this tradition of evaluating the usefulness of each rule in isolation. In addition, in contrast to some rule learners [10, 6, 23], our approach is not limited to knowledge graphs. Finally, some propositional rule-based methods used stratified rule lists, e.g. [36].

9 Conclusions

We proposed a new rule-based method for knowledge base completion, which is in many cases both more efficient and more effective than MLNs, AMIE+ and the possibilistic logic strategy from [15]. The performance of our method is due to two complementary factors. On the one hand, we showed that the use of stratified k-entailment often outperforms possibilistic logic, the one-step inference mechanism from AMIE+, and the probabilistic approach of MLNs. On the other hand, we showed that our new heuristic rule learner also produces more effective rules than AMIE+, regardless of which inference strategy is used. A closer combination of AMIE+ with our rule learning heuristic, e.g. using the latter in a post-processing step, may yield even better results, but we leave this for future work.

Acknowledgments MS acknowledges support by project no. 17-26999S granted by the Czech Science Foundation. SS has been supported by ERC Starting Grant 637277. JD is partially supported by KU Leuven Research Fund (C14/17/07) & Research Foundation - Flanders (EOS No. 30992574). OK's work has been supported by the OP VVV project CZ.02.1.01/0.0/0.0/16 019/0000765 "Research Center for Informatics" and a donation from X-Order Lab. Part of this work was done while MS and OK were already supported by the Czech Science Foundation project "Generative Relational Models" (20-19104Y).

A Sketch of Correctness and Runtime

If we had $\Gamma = \emptyset$ then the correctness of the algorithm would follow almost immediately from the same arguments that show correctness of the forward chaining procedure for classical logical reasoning. In general, when $\Gamma \neq \emptyset$, we can proceed as follows. It is easier to analyze a version of the algorithm without the filtering of non-minimal support sets. We can check that omitting this step would not affect correctness of the algorithm and that the number of iterations of this modified algorithm would not be lower than that of the full algorithm. Hence, we will analyze this simpler algorithm below.

Termination Let A be the maximum arity among the relations from $\mathcal{R}$. Let us define $N_i = |\mathcal{R}| \cdot k^{A+1} \cdot |\mathit{Incons}_i| + \sum_{a \in \mathbf{Keys}(\mathit{Support}_i)} |\mathit{Support}_i[a]|$ where Incons_i and $\mathit{Support}_i$ are states of the respective data structures at the beginning of the i-th iteration of the main loop. Then for all $i > 1$ it holds $N_{i-1} < N_i$ (in particular, this is true because we ignore filtering of non-minimal support sets). Since the set $\mathcal{C}$ is finite, there may be only a finite number of iterations of the algorithm, in particular at most $|\mathcal{R}| \cdot k^{A+1} \cdot |\mathcal{C}|^k$.

Soundness and Completeness Showing soundness and completeness for our inference algorithm in a completely rigorous way would be tedious and not really illuminating. Thus, we only provide a brief justification. Soundness is easy to check. In particular, whenever a fact h is derived using the inference algorithm, there must be a classical proof of it on a fragment of evidence, given as input, with at most k constants (this can be seen easily by inspection of the algorithm). Now, it could still be the case that h is derived from a fragment which is inconsistent with $\Phi \cup \Gamma$. However, if that was the case, this fragment would have been detected by the procedure ***FindInconsistent*** and it would have been removed together with all its occurrences among support sets stored in the hash table *Support*.

Completeness is not difficult to check either. If there is a fragment of E that is consistent with $\Phi \cup \Gamma$ and a fact h can be derived from it, then it can also be derived from it using forward chaining (this follows from the same result for classical logic). Since the fragment is consistent, neither the fragment itself nor any of its subsets can be present in the set *Incons* at any time. It follows that the fact h must be derived by the algorithm.

REFERENCES

[1] Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor, 'Hinge-loss markov random fields and probabilistic soft logic', *Journal of Machine Learning Research*, **18**(109), 1–67, (2017).

[2] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales, 'TuckER: Tensor factorization for knowledge graph completion', in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 5184–5193, (2019).

[3] Salem Benferhat, Claudette Cayrol, Didier Dubois, Jerome Lang, and Henri Prade, 'Inconsistency management and prioritized syntax-based entailment', in *IJCAI*, pp. 640–645, (1993).

[4] Salem Benferhat, Claudette Cayrol, Didier Dubois, Jérôme Lang, and Henri Prade, 'Inconsistency management and prioritized syntax-based entailment', in *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pp. 640–647, (1993).

[5] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko, 'Translating embeddings for modeling multi-relational data', in *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pp. 2787–2795, (2013).

[6] Yang Chen, Daisy Zhe Wang, and Sean Goldberg, 'ScaLeKB: scalable learning and inference over large knowledge bases', *The VLDB Journal*, **25**(6), 893–918, (2016).

[7] J. Davis, E. Burnside, I.C. Dutra, D. Page, and V. Santos Costa, 'An integrated approach to learning bayesian networks of rules', in *16th ECML*, pp. 84–95. Springer, (2005).

[8] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen, 'ProbLog: A probabilistic prolog and its application in link discovery', in *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 2462–2467, (2007).

[9] D. Dubois, J. Lang, and H. Prade, 'Possibilistic logic', in *Handbook of Logic in Artificial Intelligence and Logic Programming*, ed., D. Nute D. Gabbay, C. Hogger J. Robinson, volume 3, 439–513, Oxford University Press, (1994).

[10] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek, 'Fast rule mining in ontological knowledge bases with AMIE+', *The VLDB Journal*, **24**(6), 707–730, (2015).

[11] Alejandro J García and Guillermo R Simari, 'Defeasible logic programming: An argumentative approach', *Theory and Practice of Logic Programming*, **4**, 95–138, (2004).

[12] Hector Geffner and Judea Pearl, 'Conditional entailment: Bridging two approaches to default reasoning', *Artif. Intell.*, **53**(2-3), 209–244, (1992).

[13] Brendan Juba, 'Implicit learning of common sense for reasoning', in *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pp. 939–946, (2013).

[14] S. Kok and P. Domingos, 'Learning the structure of markov logic networks', in *Proceedings of the 22nd ICML*, pp. 441–448. ACM Press, (2005).

[15] Ondrej Kuzelka, Jesse Davis, and Steven Schockaert, 'Induction of interpretable possibilistic logic theories from relational data', in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pp. 1153–1159, (2017).

[16] Ondrej Kuzelka, Yuyi Wang, Jesse Davis, and Steven Schockaert, 'PAC-Reasoning in relational domains', in *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, pp. 927–936, (2018).

[17] John W Lloyd, *Foundations of logic programming*, Springer Science & Business Media, 2012.

[18] Loizos Michael, 'Partial observability and learnability', *Artificial Intelligence*, **174**(11), 639–669, (2010).

[19] Stephen Muggleton, 'Inverse entailment and progol', *New generation computing*, **13**(3-4), 245–286, (1995).

[20] Stephen Muggleton and Luc De Raedt, 'Inductive logic programming: Theory and methods', *The Journal of Logic Programming*, **19**, 629–679, (1994).

[21] Feng Niu, Christopher Ré, AnHai Doan, and Jude W. Shavlik, 'Tuffy: Scaling up statistical inference in markov logic networks using an RDBMS', *PVLDB*, **4**(6), 373–384, (2011).

[22] Jan Noessner, Mathias Niepert, and Heiner Stuckenschmidt, 'RockIt: Exploiting parallelism and symmetry for MAP inference in statistical relational models', in *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, (2013).

[23] Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang, 'Scalable rule learning via learning representation.', in *IJCAI*, pp. 2149–2155, (2018).

[24] Judea Pearl, 'System Z: A natural ordering of defaults with tractable applications to nonmonotonic reasoning', in *Proceedings of the 3rd Conference on Theoretical Aspects of Reasoning about Knowledge*, pp. 121–135, (1990).

[25] J. Ross Quinlan and R. Mike Cameron-Jones, 'Induction of logic programs: Foil and related systems', *New Generation Computing*, **13**(3-4), 287–312, (1995).

[26] J.R. Quinlan and R.M. Cameron, 'Induction of logic programs: FOIL and related systems', *New Generation Computing*, **13**, 287–312, (1995).

[27] Matthew Richardson and Pedro Domingos, 'Markov logic networks', *Machine Learning*, **62**, 107–136, (2006).

[28] Sebastian Riedel, 'Improving the accuracy and efficiency of MAP inference for markov logic', in *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence*, pp. 468–475, (2008).

[29] Tim Rocktäschel and Sebastian Riedel, 'End-to-end differentiable proving', in *Proceedings of the Annual Conference on Neural Information Processing Systems*, pp. 3791–3803, (2017).

[30] Mathieu Serrurier and Henri Prade, 'Introducing possibilistic logic in ILP for dealing with exceptions', *Artificial Intelligence*, **171**(16-17), 939–950, (2007).

[31] Gustav Sourek, Vojtech Aschenbrenner, Filip Zelezný, Steven Schockaert, and Ondrej Kuzelka, 'Lifted relational neural networks: Efficient learning of latent relational structures', *J. Artif. Intell. Res.*, **62**, 69–100, (2018).

[32] A. Srinivasan, *The Aleph Manual*, 2001.

[33] Théo Trouillon, Christopher R. Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard, 'Knowledge graph completion via complex tensor factorization', *J. Mach. Learn. Res.*, **18**, 130:1–130:38, (2017).

[34] Théo Trouillon, Éric Gaussier, Christopher R Dance, and Guillaume Bouchard, 'On inductive abilities of latent factor models for relational learning', *Journal of Artificial Intelligence Research*, **64**, 21–53, (2019).

[35] Leslie G. Valiant, 'Knowledge infusion', in *Proceedings of the 21st National Conference on Artificial Intelligence*, pp. 1546–1551. AAAI Press, (2006).

[36] Fulton Wang and Cynthia Rudin, 'Falling rule lists', in *Artificial Intelligence and Statistics*, pp. 1013–1022, (2015).

[37] Kaja Zupanc and Jesse Davis, 'Estimating rule quality for knowledge base completion with the relationship between coverage assumption', in *Proceedings of the Web Conference 2018*, pp. 1–9, (2018).