Implementation of a Transformation System for Relational Probabilistic Knowledge Bases Simplifying the Maximum Entropy Model Computation

Christoph Beierle, Markus Höhnerbach, Marcus Marto
University of Hagen
Hagen, Germany

Abstract
The maximum entropy (ME) model of a knowledge base $\mathcal{R}$ consisting of relational probabilistic conditionals can be defined referring to the set of all ground instances of the conditionals. The logic FO-PCL employs the notion of parametric uniformity for avoiding the full grounding of $\mathcal{R}$. We present an implementation of a rule system transforming $\mathcal{R}$ into a knowledge base that is parametrically uniform and has the same ME model, simplifying the ME model computation. The implementation provides different execution and evaluation modes, including the generation of all possible solutions.

1 Introduction
While there are several developments to extend probabilistic logic to the first-order case (Getoor and Taskar 2007), a few recent approaches employ the principle of maximum entropy (ME) (Paris 1994; Kern-Isberner 1998). One of these approaches is the logic FO-PCL (Fisseler 2010), an extension of propositional probabilistic conditional logic which combines first-order logic with probability theory to model uncertain knowledge. An example of a conditional in FO-PCL is “If $V$ likes $U$, then $U$ likes $V$ with probability 0.9, for different $U$, $V$”, formally denoted by \langle \langle \text{likes}(U,V) \mid \text{likes}(V,U) \rangle \rangle [0.9] = \{ U \neq V \}.

The models of an FO-PCL knowledge base $\mathcal{R}$ consisting of a set of such conditionals are probability distributions over possible worlds satisfying each conditional in $\mathcal{R}$, and the ME principle is used to select the uniquely determined model $\text{ME}(\mathcal{R})$ having maximum entropy. The computation of $\text{ME}(\mathcal{R})$ leads to an optimization problem with one optimization parameter to be determined for each admissible ground instance of every conditional in $\mathcal{R}$. However, if the knowledge base is parametrically uniform, i.e. all ground instances of a conditional share the same optimization parameter value, for each conditional in $\mathcal{R}$ just one optimization parameter has to be determined (Fisseler 2010). Thus, parametric uniformity significantly simplifies the task of computing $\text{ME}(\mathcal{R})$ (Finthammer and Beierle 2012).

In (Krämer and Beierle 2012), a set of transformation rules $\mathcal{PU}$ is presented allowing to transform any consistent knowledge base into a parametrically uniform knowledge base with the same maximum entropy model. In this paper, we introduce the system $\mathcal{PU}_\text{sys}$ implementing $\mathcal{PU}$ and automatically generating $\mathcal{PU}(\mathcal{R})$ for any consistent $\mathcal{R}$. This allows for a simpler ME model computation by computing $\text{ME}(\mathcal{PU}(\mathcal{R}))$ instead of $\text{ME}(\mathcal{R})$.

We very briefly sketch the basics of $\mathcal{PU}$ (Sec. 2), describe its implementation (Sec. 3), present the reasons for multiple solutions (Sec. 4) and their optimized generation (Sec. 5), give some first evaluation results and conclude (Sec. 6).

2 Interactions and Transformation Rules
In (Krämer and Beierle 2012), the reasons causing $\mathcal{R}$ to be not parametrically uniform are investigated in detail and the syntactic criterion of inter-rule and intra-rule interactions is introduced. For each of the different types of interactions, there is a corresponding interaction removing transformation rule in $\mathcal{PU}$ (cf. Figure 1). For instance, the transformation rule $(TE_1)$ removes an inter-rule interaction of type 1 by replacing a conditional $R$ with two new conditionals $\nu(\sigma(R))$ and $\nu(\overline{\sigma}(R))$, where $\sigma(R)$ is the result of applying the variable substitution $\sigma = \{V/c\}$ to $R$, and $\overline{\sigma}(R)$ is the result of adding the constraint $V \neq c$ to the constraint formula of $R$. The operator $\nu$ transforms a conditional in constraint normal form. Similarly, $(TE_2)$ and $(TE_3)$ remove inter-rule interactions of type 2 and 3. The three different types of intra-rule interactions occur within a single conditional and are removed by one of the three rules $(TA_1)$, $(TA_2)$, $(TA_3)$ in $\mathcal{PU}$ (Krämer and Beierle 2012).

Example 1 (Application of $\mathcal{PU}$) Among the conditionals
\begin{align*}
R_1 & : \langle \langle \text{likes}(U,V) \mid \text{likes}(V,U) \rangle \rangle [0.9], U \neq V \\
R_2 & : \langle \langle \text{likes}(a,V) \rangle \rangle [0.05], V \neq a
\end{align*}
there is an inter-rule interaction denoted by $R_2 \leftarrow \langle \text{likes}\rangle_{U,a} \rightarrow R_1$. $(TE_1)$ removes it by replacing $R_1$ with
\begin{align*}
R_{1,1} & : \langle \langle \text{likes}(a,V) \mid \text{likes}(V,a) \rangle \rangle [0.9], V \neq a \\
R_{1,2} & : \langle \langle \text{likes}(U,V) \rangle \rangle [0.9], U \neq V \land U \neq a.
\end{align*}

Proposition 1 (Krämer and Beierle 2012) Applying $\mathcal{PU}$ to a knowledge base $\mathcal{R}$ terminates and yields a knowledge base $\mathcal{PU}(\mathcal{R})$ having the same maximum-entropy model and $\mathcal{PU}(\mathcal{R})$ is parametrically uniform.

Due to lack of space, we refer to (Krämer and Beierle 2012; Beierle and Krämer 2014) for further details of $\mathcal{PU}$, including many examples, formal definitions and full proofs.
4 Multiple Solutions

The application of different transformation rules form \(\mathcal{PU}\) to a knowledge base \(\mathcal{R}\) may lead to different parametric uniform knowledge bases (though still having the same maximum entropy model due to Proposition 1), i.e. \(\mathcal{PU}\) is not confluent. The following knowledge base presented in (Krämer 2011) illustrates this.

Example 2 Let \(\mathcal{R} = \{R_1, R_2\}\) be the knowledge base with:

\[
R_1 : ((p(U, U) | q(V)) [0.2], U \neq V) \\
R_2 : ((p(X, Y) | q(W)) [0.3], T)
\]

There are three interactions in \(\mathcal{R}\):

\(I_a : R_1 \leftarrow p_{X,Y} \rightarrow R_2\)

\(I_b : R_1 \leftarrow (p, q)_{X,W} \rightarrow R_2\)

\(I_c : R_1 \leftarrow (p, q)_{Y,W} \rightarrow R_2\)

Choosing first the interaction \(I_a\) and applying \(\mathcal{PU}\) exhaustively yields the parametrically uniform knowledge base \(\mathcal{R}_a\) with the following four conditionals:

\[
R_1 : ((p(U, U) | q(V)) [0.2], U \neq V) \\
R_{a2} : ((p(X, Y) | q(W)) [0.3], X \neq Y) \\
R_{a3} : ((p(Y, Y) | q(V)) [0.3], T) \\
R_{a4} : ((p(Y, Y) | q(W)) [0.3], Y \neq W)
\]

Choosing first the interaction \(I_b\) and applying \(\mathcal{PU}\) exhaustively yields \(\mathcal{R}_b\) with six conditionals:

\[
R_1 : ((p(U, U) | q(V)) [0.2], U \neq V) \\
R_{b2} : ((p(Y, Y) | q(V)) [0.3], T) \\
R_{b3} : ((p(X, Y) | q(X)) [0.3], X \neq Y) \\
R_{b4} : ((p(X, Y) | q(Y)) [0.3], X \neq Y) \\
R_{b5} : ((p(Y, Y) | q(Y)) [0.3], W \neq Y) \\
R_{b6} : ((p(X, Y) | q(W)) [0.3], W \neq X \land Y \neq X \land Y)'
\]

Choosing first the interaction \(I_c\) and applying \(\mathcal{PU}\) exhaustively yields a knowledge base \(\mathcal{R}_c\) also with six conditionals; in fact, \(\mathcal{R}_c\) differs from \(\mathcal{R}_b\) only by a renaming of variables.

Thus, even when taking variable renamings into account, in Example 2, \(\mathcal{PU}\) can transform \(\mathcal{R}\) into two different parametrically uniform knowledge bases, \(\mathcal{R}_a\) and \(\mathcal{R}_b\). Here, the choice of the interaction that gets removed first determines the solution, while in general, the splitting in different solutions may occur at any stage of the transformation process.

5 Generation of All Solutions

Enumerating all solutions in a simple way by branching out every time there is more than one option which interaction to remove first, is not feasible even for small knowledge bases. It would also give no information about the number of solutions that differ in more than a variable naming. Knowledge bases obtained by \(\mathcal{PU}\) whose conditionals differ only in variable naming are equivalent. The source for this ambiguity in the transformation process is that an equivalence constraint \(A = B\) can be realized in a substitution \(A/B\) as well as \(B/A\) if \(A\) and \(B\) are both variables.

Definition 1 (pt-equivalent conditionals) Let \(\mathcal{R}\) be a knowledge base, \(R \in \mathcal{R}\), and let \(\sigma = \sigma_n \circ \ldots \circ \sigma_1\) and \(\sigma' = \sigma_m \circ \ldots \circ \sigma_1\) be substitutions obtained from applying two sequences of \(\mathcal{PU}\) transformations to \(R\). Then the

\[\mathcal{R}^\sigma = \mathcal{R}^{\sigma'}\]

1 source code of KREATOR and \(\mathcal{PU}\) can be found at http://kreator-ide.sourceforge.net/
to the knowledge base

Example 4 substitution node to said conditional node, and we do not whether a pt-equivalent conditional is already contained in equivalence between conditionals from Definition 1 to check to connect its outgoing edges. At this point we use the corresponding substitution nodes for the found interactions.

check for interactions with said nodes and add the corre-

sponding substitution nodes for the found interactions.

An algorithm to find the solutions has to make two choices during the process:

Q1: What conditionals should be checked for interactions?
Q2: Which transformations should be executed on these conditionals ensuring that all solutions are generated?

The algorithm introduced in this paper to answer Q1 and Q2 uses an auxiliary graph which is essentially a representation for the set of knowledge bases reachable through the transformation process. It is a directed graph with two types of nodes: conditional nodes representing a single conditional, and substitution nodes representing a substitution acting on a conditional. The nodes are connected such that conditional nodes are connected to their respective interaction-removing substitution nodes, and substitution nodes are connected to the conditional nodes that are the result of applying said substitution to the parent conditional.

Example 3 Fig. 2(a) is an auxiliary graph representing the solution knowledge base $R_b$ from Example 2. On the top level there are the conditionals of the original knowledge base (rectangles). Below these there are the interaction-removing substitutions $σ$ (ellipses) connected to the conditional node $R$ they apply to, and to the two resulting conditional nodes $σ(R)$ and $σ̄(R)$. Thus, each substitution node has exactly one incoming and two outgoing edges. The conditionals in $R_b$ are precisely the six leaf nodes in the graph.

Such an auxiliary graph can also be constructed for the whole transformation process behind $P∪$. The algorithm starts with the empty graph and adds a conditional node for each conditional in the initial knowledge base. Then we successively pick one conditional node, compute the set of conditional nodes in the graph that it can possibly interact with, check for interactions with said nodes and add the corresponding substitution nodes for the found interactions.

When the substitution node gets added, we also have to connect its outgoing edges. At this point we use the equivalence between conditionals from Definition 1 to check whether a pt-equivalent conditional is already contained in the graph. If this is the fact, then it suffices to connect the substitution node to said conditional node, and we do not have to add a new conditional node to the graph.

Example 4 Fig. 2(b) is the auxiliary graph corresponding to the knowledge base $R$ from Example 2. In the first row, there are the conditionals of the original knowledge base $R$, and the second row contains the substitution nodes corresponding to the three interactions $I_a, I_b, I_c$ in $R$. The third row contains the six conditionals obtained by applying the corresponding interaction removing transformations. The fourth row contains the substitution nodes corresponding to the interactions among the conditionals in the third row. Note that three of the resulting conditionals in the fifth row have multiple incoming edges since, up to pt-equivalence, they can be generated in different ways.

This operation effectively transforms the graph from a tree to a directed acyclic graph. This graph can now answer the question $Q_1$ posed before: The substitution nodes denote exactly the substitutions that can be applied to its parent conditional node during the interaction removal process.

In order to answer question $Q_2$, the auxiliary graph is reduced by identifying and removing redundancies caused by substitution nodes. For instance, let $R ∈ R$ be a conditional that has two interactions in $R$ with interaction removing substitutions $σ_1, σ_2$. Assume that those are independent, i.e. removing one interaction changes nothing about the other interaction. Then the graph will contain both $σ_1$ and $σ_2$ as substitution nodes below $R$. As these are independent from each other, $σ_2$ is also a substitution child node of $σ_1(R)$ as well as $σ_1(R)$ and vice-versa. Thus, both substitution nodes $σ_1$ and $σ_2$ below $R$ lead to the same conditionals below, and we can fuse the two substitution child nodes of $R$ to one substitution node $\{σ_1, σ_2\}$ and pick an arbitrary representative determining the edges. Removing all such redundancies in a bottom-up manner yields the reduced auxiliary graph.

Example 5 Fig. 2(c) shows the reduced graph for $R$ from Example 2. Note how there is just one conditional node with more than one substitution child node, corresponding to $R_b$.

The reduced graph can be used to determine which interaction-removing substitutions on a given conditional are sufficient for generating all solutions. Starting with the set $M$ containing the conditional nodes in the first row of the graph (i.e., the set of conditionals in the original knowledge base), do the following: While there is a conditional node $C$ in $M$ that is not a leaf node, choose (non-deterministically) one of $C$’s child substitution nodes and replace $C$ in $M$ by the two child nodes of the chosen substitution node.

Example 6 As there is only one conditional node in the reduced graph in Fig. 2(c) (i.e. $R_b$), there is only one (non-deterministic) choice to be made. Thus, the graph represents exactly the two parametrically uniform solutions $R_a$ and $R_b$ (cf. Example 2) which correspond to the leave nodes obtained by choosing either the left substitution child node $X/Y$ or the right substitution child node $X/W$ of $R_2$.

6 First Evaluation Results and Further Work $P∪_{sys}$ has been applied successfully to many different knowledge bases, including all examples given in (Fisseler 2010; Krümer and Beierle 2012; Finthammer and Beierle 2012) and a series of randomly generated examples, covering all types of interactions. The optimized generation of all solutions is much more efficient than the naive approach, e.g., generating exactly the two solutions for $R$ as in Ex. 2, compared to 28 solutions in the naive approach, or yielding all non-redundant solutions within seconds where the naive
approach does not terminate within four hours. Our current work also includes the question whether \( \mathcal{PU} \) can be modified such that a confluent set of transformation rules is obtained.

References


