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 $\langle (likes(U,V) \,|\, likes(V,U))\,[0.9]\,, U \neq V\rangle$.

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 $ME(\mathcal{R})$ having maximum entropy. The computation of $ME(\mathcal{R})$ 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 $ME(\mathcal{R})$ (Finthammer and Beierle 2012).

In (Krämer and Beierle 2012), a set of 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 $\mathcal{PU}_{sys}$ 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 $ME(\mathcal{PU}(\mathcal{R}))$ instead of $ME(\mathcal{R})$.

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 $\nu(\bar{\sigma}(R))$, where $\sigma(R)$ is the result of applying the variable substitution $\sigma = \{V/c\}$ to R, and $\bar{\sigma}(R)$ is the result of adding the constraint $V \neq c$ to the constraint formula of R. The operator ν 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*

$R_1 : \langle (likes(U,V) \,|\, likes(V,U))\,[0.9]\,, U \neq V\rangle$
$R_2 : \langle (likes(a,V))\,[0.05]\,, V \neq a\rangle$

there is an inter-rule interaction denoted by $R_2 \leftarrow \langle likes\rangle_{U,a} \rightarrow R_1$. (TE_1) *removes it by replacing* R_1 *with*

$R_{1\_1} : \langle (likes(a,V) \,|\, likes(V,a))\,[0.9]\,, V \neq a\rangle$
$R_{1\_2} : \langle (likes(U,V) \,|\, likes(V,U))\,[0.9]\,, U \neq V \wedge U \neq a\rangle$.

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.

$$(TE_1)\quad \frac{\mathcal{R} \cup \{R_1, R_2\}}{\mathcal{R} \cup \{R_1\} \cup \nu\{\sigma(R_2), \overline{\sigma}(R_2)\}} \quad \begin{array}{l} R_1 \leftarrow \langle P \rangle_{V,c} \rightarrow R_2, \\ \sigma = \{V/c\} \end{array}$$

$$(TE_2)\quad \frac{\mathcal{R} \cup \{R_1, R_2\}}{\mathcal{R} \cup \{R_1\} \cup \nu\{\sigma(R_2), \overline{\sigma}(R_2)\}} \quad \begin{array}{l} R_1 \leftarrow \langle P \rangle_{V,Z} \rightarrow R_2, \\ \sigma = \{V/Z\} \end{array}$$

$$(TE_3)\quad \frac{\mathcal{R} \cup \{R_1, R_2\}}{\mathcal{R} \cup \{R_1\} \cup \nu\{\sigma(R_2), \overline{\sigma}(R_2)\}} \quad \begin{array}{l} R_1 \leftarrow \langle P, Q \rangle_{V,Z} \rightarrow R_2, \\ \sigma = \{V/Z\} \end{array}$$

$$(TA_1)\quad \frac{\mathcal{R} \cup \{R\}}{\mathcal{R} \cup \nu\{\sigma(R), \overline{\sigma}(R)\}} \quad \begin{array}{l} \langle Q \rangle_{V,c} \rightarrow R, \\ \sigma = \{V/c\} \end{array}$$

$$(TA_2)\quad \frac{\mathcal{R} \cup \{R\}}{\mathcal{R} \cup \nu\{\sigma(R), \overline{\sigma}(R)\}} \quad \begin{array}{l} \langle Q \rangle_{V,Z} \rightarrow R, \\ \sigma = \{V/Z\} \end{array}$$

$$(TA_3)\quad \frac{\mathcal{R} \cup \{R\}}{\mathcal{R} \cup \nu\{\sigma(R), \overline{\sigma}(R)\}} \quad \begin{array}{l} \langle Q, S \rangle_{V,Z} \rightarrow R, \\ \sigma = \{V/Z\} \end{array}$$

Figure 1: Transformation rules $\mathcal{PU}$ (Krämer and Beierle 2012)

3 Implementation

The software system $\mathcal{PU}_{sys}$ implements the transformation system $\mathcal{PU}$. $\mathcal{PU}_{sys}$ has been designed as a plug-in for KREATOR[1] (Finthammer and Thimm 2012), which is an integrated development environment for relational probabilistic logic. The input knowledge base is parsed into an abstract syntax tree from which an object structure is created. The recognition of interactions and the application of the transformation rules operate directly on this structure. A transformation process can be started by executing a KREATOR script (Finthammer and Thimm 2012). All transformation parameters (e.g. transformation mode) can be set either by using a graphical user interface or within the script itself.

Transformation Modes $\mathcal{PU}_{sys}$ ensures that all conditionals of the initial knowledge base are transformed into constraint-normal form. If more than one interaction is found, one of these interactions has to be selected for the application of the corresponding transformation rule. Therefore, $\mathcal{PU}_{sys}$ offers different transformation modes for different rule application strategies.

The *Interactive* mode allows to control, monitor and trace single steps of a transformation process through a graphical user interface. In the *Automatic* mode, an applicable transformation rule is selected automatically and applied until a parametrically uniform knowledge base is reached. The *All Solutions* transformation mode creates all results that are obtainable by applying different orders of rule applications. Thereby, it avoids the multiple generation of the same parametrically uniform knowledge base, and moreover, it avoids the generation of knowledge bases that are just variants of each other with respect to variable renamings. As this mode is of particular interest when investigating properties of $\mathcal{PU}$ related e.g. to minimal solutions or confluence properties, this mode will be described in more detail in Sec. 5.

[1]Source code of KREATOR and $\mathcal{PU}_{sys}$ can be found at http://kreator-ide.sourceforge.net/

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:*
$R_1 : \langle (p(U,U) \,|\, q(V))\,[0.2]\,, U \neq V \rangle$
$R_2 : \langle (p(X,Y) \,|\, q(W))\,[0.3]\,, \top \rangle$
There are three interactions in $\mathcal{R}$:
$I_a :\ R_1 \leftarrow p_{X,Y} \rightarrow R_2$
$I_b :\ R_1 \leftarrow \langle p, q \rangle_{X,W} \rightarrow R_2$
$I_c :\ R_1 \leftarrow \langle p, q \rangle_{Y,W} \rightarrow R_2$
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:
$R_1 : \langle (p(U,U) \,|\, q(V))\,[0.2]\,, U \neq V \rangle$
$R_{a2} : \langle (p(X,Y) \,|\, q(W))\,[0.3]\,, X \neq Y \rangle$
$R_{a3} : \langle (p(Y,Y) \,|\, q(Y))\,[0.3]\,, \top \rangle$
$R_{a4} : \langle (p(Y,Y) \,|\, q(W))\,[0.3]\,, Y \neq W \rangle$
Choosing first the interaction I_b and applying $\mathcal{PU}$ exhaustively yields $\mathcal{R}_b$ with six conditionals:
$R_1 : \langle (p(U,U) \,|\, q(V))\,[0.2]\,, U \neq V \rangle$
$R_{b2} : \langle (p(Y,Y) \,|\, q(Y))\,[0.3]\,, \top \rangle$
$R_{b3} : \langle (p(X,Y) \,|\, q(X))\,[0.3]\,, X \neq Y \rangle$
$R_{b4} : \langle (p(X,Y) \,|\, q(Y))\,[0.3]\,, X \neq Y \rangle$
$R_{b5} : \langle (p(Y,Y) \,|\, q(W))\,[0.3]\,, W \neq Y \rangle$
$R_{b6} : \langle (p(X,Y) \,|\, q(W))\,[0.3]\,, W \neq X \wedge W \neq Y \wedge X \neq Y \rangle$
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* *conditionals $\sigma(R)$ and $\sigma'(R)$ are* equivalent with respect to $\mathcal{PU}$ transformations *(or just* pt-equivalent*) iff there is a variable renaming ρ such that $\rho(\sigma(R)) = \sigma'(R)$.*

Note that this notion is relative to the root conditional R. For instance, in Example 2, the two conditionals

$R_2' : \langle (p(X,X) \,|\, q(X))\,[0.3]\,, \top \rangle$
$R_2'' : \langle (p(Y,Y) \,|\, q(Y))\,[0.3]\,, \top \rangle$

originating from R_2 with the substitutions W/X and Y/X respectively W/Y and X/Y are pt-equivalent as there is $\rho = X/Y$ such that $\rho(R_2') = R_2''$.

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

Q_1 : What conditionals should be checked for interactions?

Q_2 : Which transformations should be executed on these conditionals ensuring that all solutions are generated?

The algorithm introduced in this paper to answer Q_1 and Q_2 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 is an auxiliary graph representing the solution knowledge base $\mathcal{R}_b$ 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 $\sigma(R)$ and $\bar{\sigma}(R)$. Thus, each substitution node has exactly one incoming and two outgoing edges. The conditionals in $\mathcal{R}_b$ are precisely the six leaf nodes in the graph.*

Such an auxiliary graph can also be constructed for the whole transformation process behind $\mathcal{PU}$. 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 $\mathcal{R}$ from Example 2. In the first row, there are the conditionals of the original knowledge base $\mathcal{R}$, and the second row contains the substitution nodes corresponding to the three interactions I_a, I_b, I_c in $\mathcal{R}$. 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 $R \in \mathcal{R}$ be a conditional that has two interactions in $\mathcal{R}$ 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 $\sigma_1(R)$ as well as $\bar{\sigma_1}(R)$ 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 $\{\sigma_1, \sigma_2\}$ 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 $\mathcal{R}$ from Example 2. Note how there is just one conditional node with more than one substitution child node, corresponding to R_2.*

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_2), there is only one (non-deterministic) choice to be made. Thus, the graph represents exactly the two parametrically uniform solutions $\mathcal{R}_a$ and $\mathcal{R}_b$ (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

$\mathcal{PU}_{sys}$ has been applied successfully to many different knowledge bases, including all examples given in (Fisseler 2010; Krämer 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 $\mathcal{R}$ as in Ex. 2, compared to 28 solutions in the naive approach, or yielding all non-redundant solutions within seconds where the naive

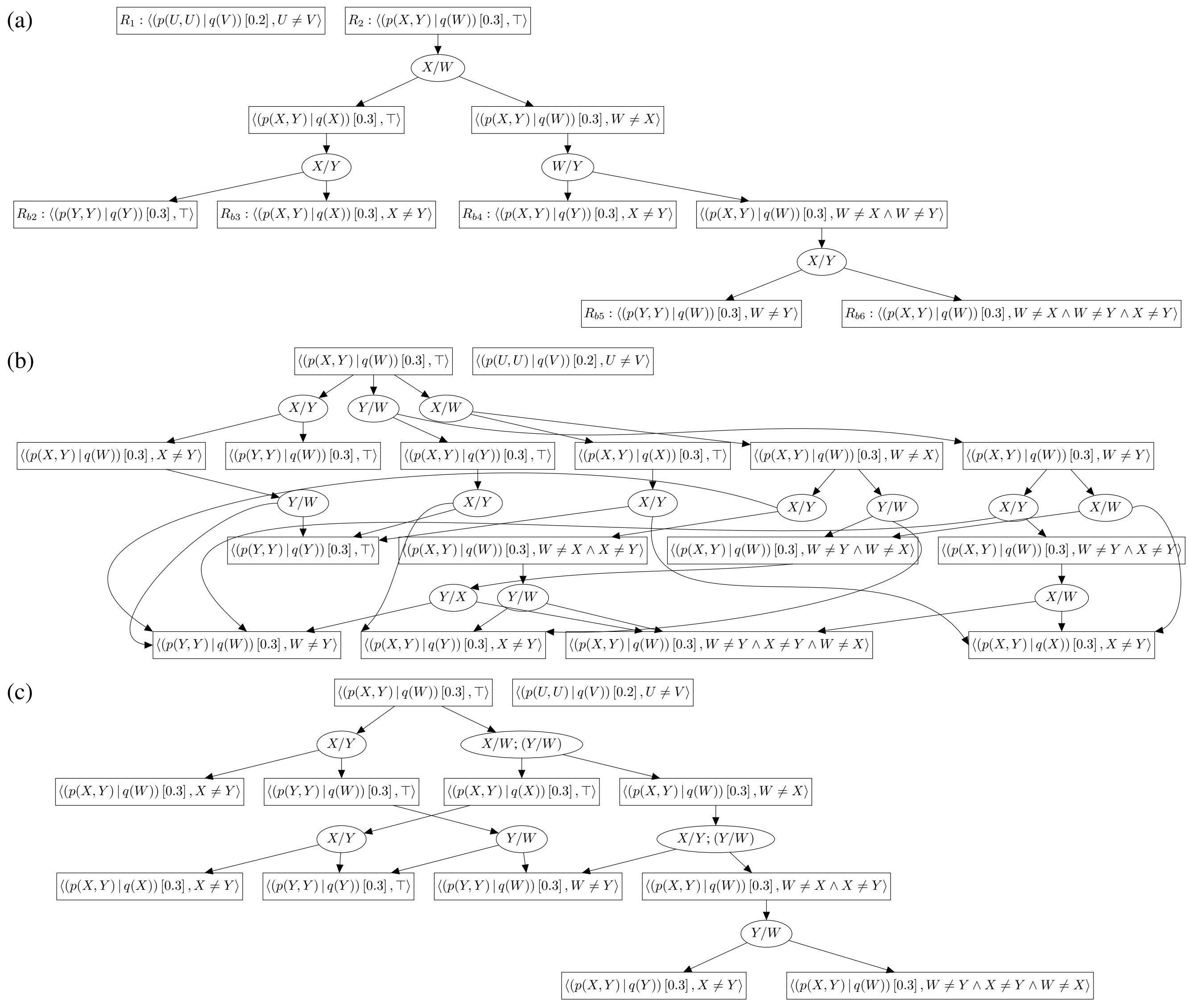
(a)
$R_1 : \langle (p(U,U) \mid q(V)) [0.2], U \neq V \rangle$
$R_2 : \langle (p(X,Y) \mid q(W)) [0.3], \top \rangle$
X/W
$\langle (p(X,Y) \mid q(X)) [0.3], \top \rangle$
$\langle (p(X,Y) \mid q(W)) [0.3], W \neq X \rangle$
X/Y
W/Y
$R_{b2} : \langle (p(Y,Y) \mid q(Y)) [0.3], \top \rangle$
$R_{b3} : \langle (p(X,Y) \mid q(X)) [0.3], X \neq Y \rangle$
$R_{b4} : \langle (p(X,Y) \mid q(Y)) [0.3], X \neq Y \rangle$
$\langle (p(X,Y) \mid q(W)) [0.3], W \neq X \wedge W \neq Y \rangle$
X/Y
$R_{b5} : \langle (p(Y,Y) \mid q(W)) [0.3], W \neq Y \rangle$
$R_{b6} : \langle (p(X,Y) \mid q(W)) [0.3], W \neq X \wedge W \neq Y \wedge X \neq Y \rangle$
(b)
$\langle (p(X,Y) \mid q(W)) [0.3], \top \rangle$
$\langle (p(U,U) \mid q(V)) [0.2], U \neq V \rangle$
X/Y
Y/W
X/W
$\langle (p(X,Y) \mid q(W)) [0.3], X \neq Y \rangle$
$\langle (p(Y,Y) \mid q(W)) [0.3], \top \rangle$
$\langle (p(X,Y) \mid q(Y)) [0.3], \top \rangle$
$\langle (p(X,Y) \mid q(X)) [0.3], \top \rangle$
$\langle (p(X,Y) \mid q(W)) [0.3], W \neq X \rangle$
$\langle (p(X,Y) \mid q(W)) [0.3], W \neq Y \rangle$
Y/W
X/Y
X/Y
X/Y
Y/W
X/Y
X/W
$\langle (p(Y,Y) \mid q(Y)) [0.3], \top \rangle$
$\langle (p(X,Y) \mid q(W)) [0.3], W \neq X \wedge X \neq Y \rangle$
$\langle (p(X,Y) \mid q(W)) [0.3], W \neq Y \wedge W \neq X \rangle$
$\langle (p(X,Y) \mid q(W)) [0.3], W \neq Y \wedge X \neq Y \rangle$
Y/X
Y/W
X/W
$\langle (p(Y,Y) \mid q(W)) [0.3], W \neq Y \rangle$
$\langle (p(X,Y) \mid q(Y)) [0.3], X \neq Y \rangle$
$\langle (p(X,Y) \mid q(W)) [0.3], W \neq Y \wedge X \neq Y \wedge W \neq X \rangle$
$\langle (p(X,Y) \mid q(X)) [0.3], X \neq Y \rangle$
(c)
$\langle (p(X,Y) \mid q(W)) [0.3], \top \rangle$
$\langle (p(U,U) \mid q(V)) [0.2], U \neq V \rangle$
X/Y
$X/W; (Y/W)$
$\langle (p(X,Y) \mid q(W)) [0.3], X \neq Y \rangle$
$\langle (p(Y,Y) \mid q(W)) [0.3], \top \rangle$
$\langle (p(X,Y) \mid q(X)) [0.3], \top \rangle$
$\langle (p(X,Y) \mid q(W)) [0.3], W \neq X \rangle$
X/Y
Y/W
$X/Y; (Y/W)$
$\langle (p(X,Y) \mid q(X)) [0.3], X \neq Y \rangle$
$\langle (p(Y,Y) \mid q(Y)) [0.3], \top \rangle$
$\langle (p(Y,Y) \mid q(W)) [0.3], W \neq Y \rangle$
$\langle (p(X,Y) \mid q(W)) [0.3], W \neq X \wedge X \neq Y \rangle$
Y/W
$\langle (p(X,Y) \mid q(Y)) [0.3], X \neq Y \rangle$
$\langle (p(X,Y) \mid q(W)) [0.3], W \neq Y \wedge X \neq Y \wedge W \neq X \rangle$

Figure 2: (a) Auxiliary graph for $\mathcal{R}_b$, (b) auxiliary graph for $\mathcal{R}$, and (c) reduced auxiliary graph for $\mathcal{R}$ from Example 2

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

Beierle, C., and Krämer, A. 2014. Achieving parametric uniformity for knowledge bases in a relational probabilistic conditional logic with maximum entropy semantics. *Annals of Mathematics and Artificial Intelligence*. (to appear).

Finthammer, M., and Beierle, C. 2012. How to exploit parametric uniformity for maximum entropy reasoning in a relational probabilistic logic. In *JELIA 2012*, volume 7519 of *LNAI*, 189–201. Springer.

Finthammer, M., and Thimm, M. 2012. An integrated development environment for probabilistic relational reasoning. *Logic Journal of the IGPL* 20(5):831–871.

Fisseler, J. 2010. *Learning and Modeling with Probabilistic Conditional Logic*, volume 328. Amsterdam: IOS Press.

Getoor, L., and Taskar, B., eds. 2007. *Introduction to Statistical Relational Learning*. MIT Press.

Kern-Isberner, G. 1998. Characterizing the principle of minimum cross-entropy within a conditional-logical framework. *Artificial Intelligence* 98:169–208.

Krämer, A., and Beierle, C. 2012. On lifted inference for a relational probabilistic conditional logic with maximum entropy semantics. In *FoIKS 2012*, volume 7153 of *LNCS*, 224–243. Springer.

Krämer, A. 2011. Transformation rules for lifted inference in relational probabilistic logic knowledge bases. B.Sc. Thesis, FernUniversität in Hagen, Germany.

Paris, J. 1994. *The uncertain reasoner's companion – A mathematical perspective*. Cambridge University Press.