Generation of Parametrically Uniform Knowledge Bases in a Relational Probabilistic Logic with Maximum Entropy Semantics

Christoph Beierle, Markus Höhnerbach, Marcus Marto

Dept. of Computer Science, University of Hagen, Germany

Abstract. In a relational setting, the maximum entropy model of a set of probabilistic conditionals can be defined referring to the full set of ground instances of the conditionals. The logic FO-PCL uses the notion of parametric uniformity to ensure that the full grounding of the conditionals can be avoided, thereby greatly simplifying the maximum entropy model computation. In this paper, we describe a system that realises an approach transforming an FO-PCL knowledge base consisting of relational probabilistic conditionals into a knowledge base having the same maximum entropy model that is parametrically uniform. The implemented system provides different execution and evaluation modes, including the generation of all possible solutions, and is available within an integrated development environment for relational probabilistic logic.

1 Introduction

There are different approaches addressing the challenge of modelling uncertainty by combining logic with probabilities, see e.g. the foundational work reported in [18, 6, 7]. Probabilistic conditional logic considers conditional probabilities $P(B|A)$ for conditionals of the form *if A then B with probability x* [1, 19], formally denoted by $(B|A)[x]$; thus, $P(B|A)$ denotes the probability of B given that we know A holds. While these approaches are often based upon propositional logic, there are also several developments aiming at exploiting the expressiveness of relational or general first order logic in combination with probabilities, e.g. Bayesian logic programs and Markov logic networks (see [11] for an overview). Some of these approaches (see e.g. [15, 22]) employ the principle of maximum entropy (ME) [20, 14]. Among these, there is also FO-PCL [10], a first-order extension of propositional probabilistic conditional logic. 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 $\langle (likes(U,V) \mid likes(V,U))\,[0.9]\,, U \neq V \rangle$.

The models of an FO-PCL knowledge base $\mathcal{R}$ consisting of a set of such conditionals are probability distributions over possible worlds [12] satisfying each conditional in $\mathcal{R}$, where a possible world is a subset of the Herbrand base which is determined by the set of all ground instances of conditionals satisfying the respective constraint. 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 optimisation problem with one optimisation 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 optimisation parameter value, for *each conditional* in $\mathcal{R}$ just one optimisation parameter has to be determined [10]. Thus, parametric uniformity significantly simplifies the task of computing $ME(\mathcal{R})$ [8].

In [17, 4], 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 $ME(\mathcal{R})$.

This paper is an extended version of [3]. In particular, we expand the background about FO-PCL by recalling the notions of inter-rule und intra-rule intercations, give a more detailed overview on $\mathcal{PU}_{sys}$, and provide results obtained from an evaluation of the system. After recalling the basics of FO-PCL and $\mathcal{PU}$ (Sec. 2), we present $\mathcal{PU}_{sys}$ in Sec. 3. We illustrate the reasons for multiple solutions of $\mathcal{PU}$ transformations (Sec. 4), describe their optimised generation (Sec. 5), give some first application and evaluation results (Sec. 6), and conclude and point out further work (Sec. 7).

2 Background: Interactions and $\mathcal{PU}$ Transformation Rules

An FO-PCL conditional $\langle(\psi \mid \phi)\,[\xi]\,, C\rangle$ consists of an antecedent ϕ and a consequent ψ, both being quantifier-free first-order, non-equational formulas, a probability value $\xi \in [0, 1]$, and a constraint formula C with inquatations over the variables (and constants) occurring in ϕ and ψ. As an illustration of a simple FO-PCL knowledge base we consider the following:

Example 1 (Misanthrope). The misanthrope example (adapted from [10]) models the friendship relations for a group of people. Usually, if a person V likes a person U, then U also likes V. However, there is a misanthrope *a*, and the chances that *a* likes someone is considerably low. The knowledge base $\mathcal{R}_{MI} = \{R_1, R_2\}$ contains two conditionals:

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

A ground instance of a conditional satisfying the conditional's constraint formula like $U \neq V$ is called *admissible*. A model of an FO-PCL knowledge base $\mathcal{R}$ is a joint probability function P satisfying each conditional in $\mathcal{R}$ where P satisfies a conditional $\langle(\psi \mid \phi)\,[\xi]\,, C\rangle$ iff $P(\psi_g|\phi_g) = \xi$ for every admissible ground instance $\langle(\psi_g \mid \phi_g)\,[\xi]\,, \top\rangle$ of the conditional [10]. Note that a full grounding of $\mathcal{R}_{MI}$ ignoring the constraint formulas would immediately yield an inconsistency since there is no probability distribution P with $P(likes(a, a)) = 0.05$ and $P(likes(a, a)|likes(a, a)) = 0.9$. Parametric uniformity [10] significantly simplifies the task of computing the maximum entropy [20, 14] model $ME(\mathcal{R})$ since it reduces the number of optimisation parameters to be taken into account from the number of admissible ground instances to the number of conditionals in $\mathcal{R}$ [8]. For details about the underlying optimisation problem and its parameters determining the FO-PCL model $ME(\mathcal{R})$, we refer to [10, 8].

In [17], the reasons causing $\mathcal{R}$ to be not parametrically uniform are investigated in detail. A central observation is that for any parametrically uniform $\mathcal{R}$, the sets of ground

instances of atoms and pairs of different atoms occurring in the admissible ground instances of two conditionals must be either disjoint or identical (otherwise there is an *imbalanced sharing* [17, 4]). For instance, $\mathcal{R}_{MI}$ from Ex. 1 is not parametrically uniform since the sets of admissible ground atoms originating from $likes(U, V)$ of conditional R_1 and from $likes(a, V)$ of conditional R_2 are neither equal nor disjoint. For a single conditional it is required that each admissible ground instance of an atom p or of a pair of different atoms p', p'' occurs the same number of times in the ground instances of the conditional as any other admissible ground instance of the atom p or of the pair of atoms p', p'' (otherwise there is an *imbalanced use* [17, 4]). The syntactic criterion of *inter-rule* and *intra-rule interactions* [17] identifies these constellations by analysing the syntactic structure of the conditionals in $\mathcal{R}$; no grounding operation is required.

Inter-Rule Interactions An *inter-rule interaction of type 1* between two conditionals R_1 and R_2 with respect to P, regarding the variable V and the constant c, denoted $R_1 \leftarrow \langle P \rangle_{V,c} \rightarrow R_2$ is a situation

a) $R_1 = \langle (\dots P(\dots, c, \dots) \dots)[\xi_{R_1}], C_{R_1} \rangle$,
$R_2 = \langle (\dots P(\dots, V, \dots) \dots)[\xi_{R_2}], C_{R_2} \rangle$, and $C_{R_2} \nvDash V \neq c$, or
b) $R_1 = \langle (\dots P(\dots, U, \dots) \dots)[\xi_{R_1}], C_{R_1} \rangle$, $C_{R_1} \vDash U \neq c$,
$R_2 = \langle (\dots P(\dots, V, \dots) \dots)[\xi_{R_2}], C_{R_2} \rangle$, and $C_{R_2} \nvDash V \neq c$.

where $C_{R_2} \nvDash V \neq c$ denotes that from C_{R_2}, it does not follow that V is not equal to c.

An *inter-rule interaction of type 2* between R_1 and R_2 with respect to P, regarding the variables V and Z, denoted $R_1 \leftarrow \langle P \rangle_{V,Z} \rightarrow R_2$, is a situation with

a) $R_1 = \langle (\dots P(\dots, U, \dots, U, \dots) \dots)[\xi_{R_1}], C_{R_1} \rangle$,
$R_2 = \langle (\dots P(\dots, V, \dots, Z, \dots) \dots)[\xi_{R_2}], C_{R_2} \rangle$, and $C_{R_2} \nvDash V \neq Z$, or
b) $R_1 = \langle (\dots P(\dots, U, \dots, W, \dots) \dots)[\xi_{R_1}], C_{R_1} \rangle, C_{R_1} \vDash U \neq W$,
$R_2 = \langle (\dots P(\dots, V, \dots, Z, \dots) \dots)[\xi_{R_2}], C_{R_2} \rangle$, and $C_{R_2} \nvDash V \neq Z$.

An *inter-rule interaction of type 3* between R_1 and R_2 with respect to P and Q, regarding the variables V and Z, denoted $R_1 \leftarrow \langle P, Q \rangle_{V,Z} \rightarrow R_2$, involves two different atoms $P(\dots)$ and $Q(\dots)$ and is a situation with

a) $R_1 = \langle (\dots P(\dots, U, \dots) \dots Q(\dots, U, \dots) \dots)[\xi_{R_1}], C_{R_1} \rangle$,
$R_2 = \langle (\dots P(\dots, V, \dots) \dots Q(\dots, Z, \dots) \dots)[\xi_{R_2}], C_{R_2} \rangle$, and $C_{R_2} \nvDash V \neq Z$, or
b) $R_1 = \langle (\dots P(\dots, U, \dots) \dots Q(\dots, W, \dots) \dots)[\xi_{R_1}], C_{R_1} \rangle, C_{R_1} \vDash U \neq W$,
$R_2 = \langle (\dots P(\dots, V, \dots) \dots Q(\dots, Z, \dots) \dots)[\xi_{R_2}], C_{R_2} \rangle$, and $C_{R_2} \nvDash V \neq Z$.

The three types of inter-rule interactions capture exactly the different reasons for imbalanced sharing [17]. E.g., for $\mathcal{R}_{MI}$ the imbalanced sharing sketched above is identified by the inter-rule interaction of type 1 $R_2 \leftarrow \langle likes \rangle_{U,a} \rightarrow R_1$.

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. The application of TE_1 replaces 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, a normal form required for the recognition of

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

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

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

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

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

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

Fig. 1. Transformation rules $\mathcal{PU}$ [17]

interactions. Similarly, (TE_2) and (TE_3) remove inter-rule interactions of type 2 and 3, respectively.

Example 2 (Application of $\mathcal{PU}$ to $\mathcal{R}_{MI}$). The knowledge base $\mathcal{R}_{MI}$ of Ex. 1 is not parametrically uniform since it contains two inter-rule interactions of type 1 denoted by $R_2 \leftarrow \langle likes \rangle_{U,a} \rightarrow R_1$ and $R_2 \leftarrow \langle likes \rangle_{V,a} \rightarrow R_1$. The application of TE_1 for the first interaction replaces R_1 with

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

R_{1_1} is the result of the substitution $\sigma = \{U/a\}$ applied to R_1 and R_{1_2} is the result of adding the constraint $U \neq a$ to R_1. The atom $likes(U, V)$ of R_1 that caused the first interaction becomes $likes(a, V)$ in R_{1_1} and its set of ground atoms is now identical to the set of ground atoms from $likes(a, V)$ of R_2. The added constraint $U \neq a$ in R_{1_2} leads to disjoint sets of the ground atoms of the discussed atoms. The second interaction between R_2 and R_{1_2} is also removed by applying TE_1, thereby replacing R_{1_2} with

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

The resulting knowledge base is $\mathcal{PU}(\mathcal{R}_{MI}) = \{R_{1\_1},\ R_{1\_2\_1},\ R_{1\_2\_2},\ R_2\}$. □

Intra-Rule Interactions While inter-rule interactions involve two conditionals, intra-rule interactions occur within a single conditional and capture the notion of imbalanced use. Suppose R is a conditional containing atoms P_R, Q_R with the predicate symbols P and Q, respectively.

An *intra-rule interaction of type 1* in R with respect to Q, regarding the variable V and the constant c, denoted $\langle Q \rangle_{V,c} \rightarrow R$, is a situation with $P_R = P(\ldots, U, \ldots),\ Q_R = Q(\ldots, V, \ldots), U \notin vars(Q_R),\ C_R \vDash U \neq V,\ C_R \vDash U \neq c$, and $C_R \nvDash V \neq c$.

An *intra-rule interaction of type 2* in R with respect to Q, regarding the variables V and Z, denoted $\langle Q \rangle_{V,Z} \rightarrow R$, is a situation with $P_R = P(\ldots, U, \ldots)$, $Q_R = Q(\ldots, V, \ldots, Z \ldots)$, $U \notin vars(Q_R)$, $C_R \vDash U \neq V$, $C_R \vDash U \neq Z$, and $C_R \nvDash V \neq Z$.

If S_R with predicate symbol S is a further atom in R, an *intra-rule interaction of type 3* in R with respect to Q and S, regarding the variables V and Z, denoted $\langle Q, S \rangle_{V,Z} \rightarrow R$, is a situation with $P_R = P(\ldots, U, \ldots)$, $Q_R = Q(\ldots, V, \ldots)$, $S_R = S(\ldots Z \ldots)$, $U \notin vars(Q_R)$, $U \notin vars(S_R)$, $C_R \vDash U \neq V$, $C_R \vDash U \neq Z$, and $C_R \nvDash V \neq Z$.

Each type of intra-rule interaction can be removed by one of the three rule (TA_1), (TA_2), (TA_3) in $\mathcal{PU}$ (Figure 1).

Example 3 (Intra-rule interaction of type 1). Suppose that the signature for the knowledge base $\mathcal{R}_{IA1} = \{R_1\}$ contains the constants $\mathcal{D} = \{a, b, c\}$:

$$R_1 : \langle (p(U) \,|\, q(V)) \, [\xi] \, , U \neq V \wedge U \neq a \rangle$$

There is an intra-rule interaction of type 1 denoted $\langle q \rangle_{V,a} \rightarrow R_1$. The reason for this interaction is that the number of occurrences of the ground atom $q(a)$ in the admissible ground instances of R_1 is 2, but for $q(b)$ (and also $q(c)$) only 1. Note that this imbalance persists for any larger set of constants $\mathcal{D}$. The application of (TA_1) replaces R_1 with

$$R_{1\_1} : \langle (p(U) \,|\, q(a)) \, [\xi] \, , U \neq a \rangle$$
$$R_{1\_2} : \langle (p(U) \,|\, q(V)) \, [\xi] \, , U \neq V \wedge U \neq a \wedge V \neq a \rangle$$

and the resulting knowledge base $\mathcal{PU}(\mathcal{R}_{IA1})$ is parametrically uniform. □

The importance of inter- and intra-rule interactions is that they fully capture the reasons for a knowledge base to be not parametrically uniform. Correctness and completeness of the set of interaction removing transformation rules $\mathcal{PU}$ is given by the following proposition:

Proposition 1. [17] *Exhaustively applying $\mathcal{PU}$ to a knowledge base $\mathcal{R}$ yields a knowledge base $\mathcal{PU}(\mathcal{R})$ such that $\mathcal{R}$ and $\mathcal{PU}(\mathcal{R})$ have the same maximum-entropy model and $\mathcal{PU}(\mathcal{R})$ is parametrically uniform.*

A proof of this proposition and further details of $\mathcal{PU}$ can be found in [17, 4]. As a consequence of Proposition 1, we can obtain the maximum entropy model $ME(\mathcal{R})$ by computing $ME(\mathcal{PU}(\mathcal{R}))$, thereby reducing the number of optimisation parameters to be considered from the number of admissible ground instances of $\mathcal{R}$ to the number of conditionals in $\mathcal{PU}(\mathcal{R})$. For instance, considering a set D of constants having 15 elements in Ex. 1, there are 224 admissible ground instances for $\mathcal{R}_{MI}$, but only 4 conditionals in $\mathcal{PU}(\mathcal{R}_{MI})$ (cf. Ex. 2).

3 Implementation of the Transformation System $\mathcal{PU}$

In this section, we present an overview of the software system $\mathcal{PU}_{sys}$ that implements the transformation system $\mathcal{PU}$. $\mathcal{PU}_{sys}$ has been designed as a plugin for KReator[1] [9],

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

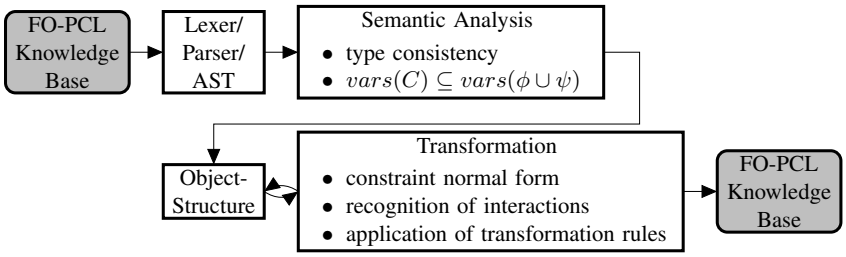

Fig. 2. Conceptual pipeline of the system $\mathcal{PU}_{sys}$.

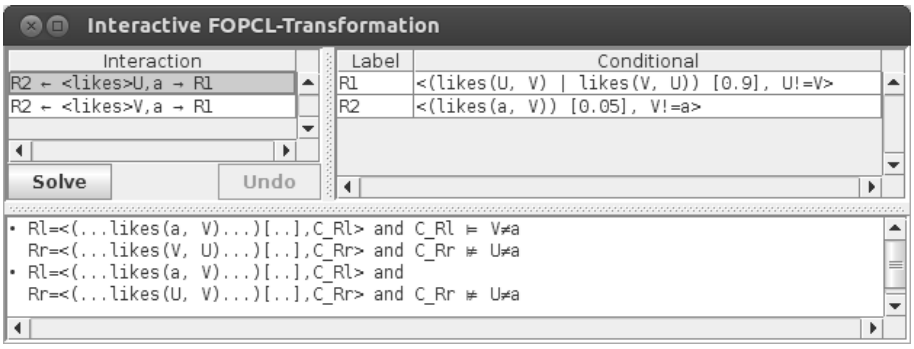

Fig. 3. Interactive transformation mode applied to Ex. 1.

which is an integrated development environment for relational probabilistic logic. The basic architecture of the implemented transformation system is illustrated in Figure 2. The input knowledge base is parsed into an abstract syntax tree (AST) from which an object structure is created. Thereby it is ensured that the types of variables and constants in a conditional are conform with the signature of the knowledge base (type consistency) and that the variables of the constraint formula are used in either the premise or the conclusion of the conditional. The key operations of the transformation phase are the transformation of conditionals in constraint normal form, the recognition of interactions, and the application of the transformation rules. These operations operate directly on the object structure. After the transformation phase, the output knowledge base is created from the resulting object structure.

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; moreover, $\mathcal{PU}_{sys}$ can be easily extended with additional transformation modes. The *Interactive* mode allows to control, monitor and trace single steps of a transformation process through a graphical user interface. All interactions present in a knowledge base are listed in a short form notation. They can be selected separately to apply the corresponding transformation rule or to view more detailed information in a higher-level, declarative notation which originates from the

definition of the interactions. In the *Automatic* mode, an applicable transformation rule is selected automatically and applied, and this step is repeated 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.

User Interface A transformation process can be started with KReator by executing a KReator-Script [9]. All transformation parameters (e.g. transformation mode) can be set either by using a graphical user interface or within the script itself. The transformation systems also supports batch processing for multiple knowledge base files. A screenshot of the *Interactive* transformation mode applied to $\mathcal{R}_{MI}$ from Ex. 1 is shown in Figure 3. The right table displays the conditionals of the knowledge base to be transformed and is updated accordingly after a transformation rule has been applied. All interactions that have been recognised are listed in a short form notation in the left table. They can be selected to apply the corresponding transformation rule. The bottom pane displays detailed information about the reasons why an interaction exists by tracing it back to the declarative notation used in the definitions of the interactions.

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 [16] illustrates this.

Example 4. 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 4, $\mathcal{PU}$ 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. From a computational point of view, one is especially interested in small knowledge bases. Since it is not clear which particular choice of interaction removal will lead to a minimal knowledge base, such a minimal knowledge base can be obtained by selecting it from the set of all solutions.

5 Generation of all Solutions

In principle, it is possible to enumerate the solutions in a simple way by branching out the search algorithm every time that there is more than one option which interaction to remove first. However, this might be not feasible even for small knowledge bases. It would also give no information about the number of unique solutions, i.e. solutions that differ in more than a variable naming.

Knowledge bases obtained by $\mathcal{PU}$ whose conditionals differ only in variable naming are equivalent. A source for this ambiguity in the transformation process is that equality constraints are realised using substitutions. However, an equality constraint $A = B$ can be realised by a substitution A/B as well by 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 4,

$$R'_2 : \langle (p(X,X) \,|\, q(X))\,[0.3]\,, \top \rangle$$
$$R''_2 : \langle (p(Y,Y) \,|\, q(Y))\,[0.3]\,, \top \rangle$$

originating from R_2 with the substitutions W/X and Y/X respectively W/Y and X/Y are pt-equivalent as there is the variable renaming $\rho = X/Y$ such that $\rho(R'_2) = R''_2$.

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

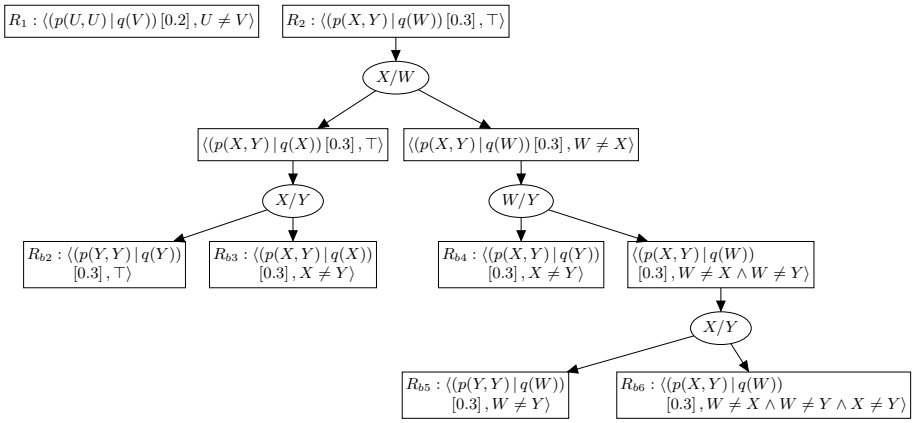

Fig. 4. The auxiliary graph of the solution knowledge base $\mathcal{R}_b$ from Example 4.

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?

A naive approach might choose all conditionals and all transformations, leading to an algorithm that branches out extremely and that may generate a huge number of knowledge bases. The algorithm introduced in this paper uses an auxiliary graph to answer those two questions in a more sensible way. An auxiliary graph is essentially a representation for the set of knowledge bases that can be reached through the transformation process. It is a directed graph containing 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 5. Fig. 4 is an auxiliary graph representing the solution knowledge base $\mathcal{R}_b$ from Example 4. It shows how such graphs look like: 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 $\bar{\sigma}(R)$. This means that each substitution node has exactly one incoming and two leaving 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 6. Fig. 5 is the auxiliary graph corresponding to the knowledge base $\mathcal{R}$ from Example 4. In the first row, there are the two 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. We will give an exemplary situation of removing redundancies introduced by the order of interaction removal for independent interactions on the same conditional: 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. This means that we can fuse the two substitution child nodes of R to one substitution node $\{\sigma_1, \sigma_2\}$. We can pick an arbitrary *representative* that will represent the substitution node and especially determine the edges. Removing all such redundancies in a bottom-up manner yields the *reduced* auxiliary graph which is uniquely determined if a choice function for picking a representative for fused substitution nodes is given.

Example 7. Fig. 6 shows the reduced graph to Example 4. 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. Using the reduced graph in this way allows the generation of all solutions without getting multiple copies or variable renamed versions.

Example 8. As there is only one conditional node in the reduced graph in Fig. 6 (i.e. R_2), there is only one (non-deterministic) choice to be made. Thus, the graph represents

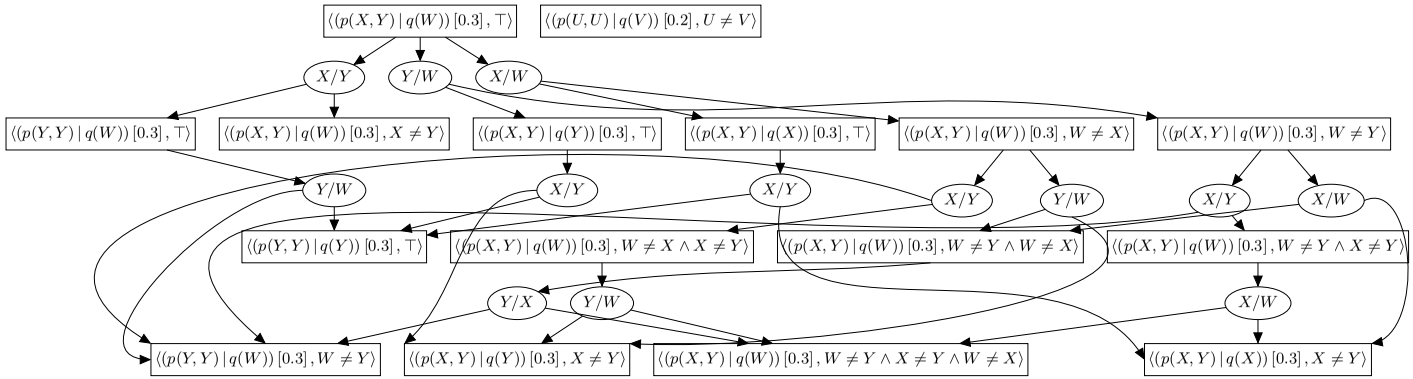

Fig. 5. The auxiliary graph of Example 4

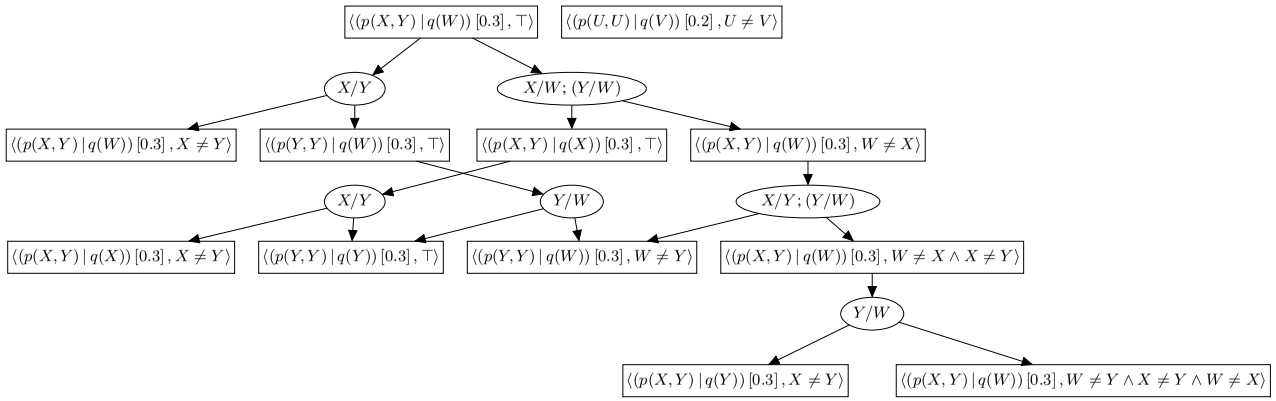

Fig. 6. Reduced graph of the auxiliary graph from Example 4

exactly the two parametrically uniform solutions $\mathcal{R}_a$ and $\mathcal{R}_b$ (cf. Example 4) 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 Applications and First Evaluation Results

The system $\mathcal{PU}_{sys}$ was successfully applied to many different examples, including all examples discussed in [10, 17, 8, 4]. As expected, the optimised generation of all solutions presented in Sec. 5 turned out to be much more efficient than an naive approach of simply executing all possible transformation sequences. For instance, for the knowledge base $\mathcal{R}$ (Example 4) exactly the two solutions were generated, compared to 28 knowledge bases in the naive approach. There are also examples, where the naive approach generates many solutions (e.g. 96), while the optimised version yields a single unique solution [2]. For another knowledge base, all four (non-redundant) solutions were generated within 10 seconds, while the naive approach did not terminate within more than four hours. In [2], $\mathcal{PU}_{sys}$ is tested and evaluated with respect to many different knowledge bases, including a series of synthetically generated knowledge bases varying in the number of sorts, constants, predicates, predicate arities, and conditionals. Among these parameters, especially increasing the arity of predicates increased the number of conditionals in $\mathcal{PU}(\mathcal{R})$ due to an increased number of interactions in the original knowledge base $\mathcal{R}$. For all knowledge bases considered in [10, 17, 8, 4], the increase from $|\mathcal{R}|$ to $|\mathcal{PU}(\mathcal{R})|$ was at most by a factor of 2.2, while for some synthetically generated knowledge bases there was a factor of 8.1. However, as pointed out in Sec. 2, crucial for maximum entropy computation is the relationship between $|\mathcal{PU}(\mathcal{R})|$ and $|\operatorname{gnd}(\mathcal{R})|$ (i.e. the number of admissible groundings of $\mathcal{R}$); in the evaluation in [2], this relationship is often characterised by factors in the order of magnitude between 10^2 and 10^4. As shown in [8], this factor between $|\mathcal{PU}(\mathcal{R})|$ and $|\operatorname{gnd}(\mathcal{R})|$ is responsible for the simplification obtained in computing $ME(\mathcal{R})$ by determining $ME(\mathcal{PU}(\mathcal{R}))$ instead.

7 Conclusions and Further Work

The software system $\mathcal{PU}_{sys}$ implements the transformation system $\mathcal{PU}$ and provides an optimised generation of all possible solutions, thereby transforming any consistent FO-PCL knowledge $\mathcal{R}$ base to a semantically equivalent and parametrically uniform one and thus simplifying the maximum entropy model computation. An open question is how $\mathcal{PU}$ should be modified so that a confluent set of transformation rules is obtained. While the objective of $\mathcal{PU}$ is to simplify the ME model computation, thereby not taking inference into account, the techniques used in $\mathcal{PU}$ are related to techniques like shattering used in lifted inference [21, 5, 13] Our current work also includes the development of sophisticated methods of using the obtained maximum entropy model $ME(\mathcal{R})$ for deriving the probability $ME(\mathcal{R})(F)$ for a given formula or conditional F; here, techniques of lifted inference might be employed.

References

1. E. Adams. *The Logic of Conditionals*. D. Reidel, Dordrecht, 1975.

2. A. Becker. Test, evaluation and assessment of a transformation system for knowledge bases with relational probabilistic conditionals. M.Sc. Thesis, Dept. of Computer Science, University of Hagen, Germany, 2014. (in German).
3. C. Beierle, M. Höhnerbach, and M. Marto. Implementation of a transformation system for relational probabilistic knowledge bases simplifying the maximum entropy model computation. In *Proc. FLAIRS 2014*, pages 486–489, Menlo Park, CA, 2014. AAAI Press.
4. C. Beierle and A. Krämer. Achieving parametric uniformity for knowledge bases in a relational probabilistic conditional logic with maximum entropy semantics. *Annals of Mathematics and Artificial Intelligence*, 2014. (to appear).
5. R. de Salvo Braz, E. Amir, and D. Roth. Lifted first-order probabilistic inference. In L. P. Kaelbling and A. Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1319–1325. Professional Book Center, 2005.
6. R. Fagin, J. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87:78–128, 1990.
7. R. Fagin and J. Y. Halpern. Reasoning about knowledge and probability. *J. ACM*, 41(2):340–367, 1994.
8. M. Finthammer and C. Beierle. How to exploit parametric uniformity for maximum entropy reasoning in a relational probabilistic logic. In L. Fariñas del Cerro, A. Herzig, and J. Mengin, editors, *JELIA*, volume 7519 of *LNAI*, pages 189–201. Springer, 2012.
9. M. Finthammer and M. Thimm. An integrated development environment for probabilistic relational reasoning. *Logic Journal of the IGPL*, 20(5):831–871, 2012.
10. J. Fisseler. *Learning and Modeling with Probabilistic Conditional Logic*, volume 328 of *Dissertations in Artificial Intelligence*. IOS Press, Amsterdam, 2010.
11. L. Getoor and B. Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, 2007.
12. J. Halpern. *Reasoning About Uncertainty*. MIT Press, 2005.
13. A. Jaimovich, O. Meshi, and N. Friedman. Template based inference in symmetric relational Markov random fields. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2007.
14. G. Kern-Isberner. Characterizing the principle of minimum cross-entropy within a conditional-logical framework. *Artificial Intelligence*, 98:169–208, 1998.
15. G. Kern-Isberner and T. Lukasiewicz. Combining probabilistic logic programming with the power of maximum entropy. *Artificial Intelligence, Special Issue on Nonmonotonic Reasoning*, 157(1-2):139–202, 2004.
16. A. Krämer. Transformation rules for lifted inference in relational probabilistic logic knowledge bases. B.Sc. Thesis, Dept. of Computer Science, University of Hagen, Germany, 2011.
17. A. Krämer and C. Beierle. On lifted inference for a relational probabilistic conditional logic with maximum entropy semantics. In T. Lukasiewicz and A. Sali, editors, *Foundations of Information and Knowledge Systems (FoIKS 2012)*, volume 7153 of *LNCS*, pages 224–243. Springer, 2012.
18. N. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.
19. D. Nute and C. Cross. Conditional logic. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic*, volume 4, pages 1–98. Kluwer Academic Publishers, second edition, 2002.
20. J. Paris. *The uncertain reasoner's companion – A mathematical perspective*. Cambridge University Press, 1994.
21. D. Poole. First-order probabilistic inference. In G. Gottlob and T. Walsh, editors, *Proc. IJCAI-03*, pages 985–991. Morgan Kaufmann, 2003.
22. M. Thimm, G. Kern-Isberner, and J. Fisseler. Relational probabilistic conditional reasoning at maximum entropy. In *ECSQARU*, volume 6717 of *LNCS*, pages 447–458. Springer, 2011.