ContraBovemRufum: A System for Probabilistic Lexicographic Entailment *

Tobias Henrik Näth[1], Ralf Möller[1]

Hamburg University of Technology,
Institute of Software Systems,
21079 Hamburg, Germany,
{r.f.moeller, tobias.naeth}@tu-harburg.de

Abstract. Representing probabilistic knowledge in combination with a description logic has been a research topic for quite some time. In [1] one of such combinations is introduced. We present our implementation of the proposed system and some modeling observations we made.

1 Introduction

For at least ten years, modeling of uncertainty combined with description logics has been a topic of research. First approaches of combinations of description logic and probabilities are described in [2–4]. Recently, techniques for so-called probabilistic lexicographic reasoning services were proposed by Giugno and Lukasiewicz [1]. Their approach, named P-$\mathcal{SHOQ}(D)$, reduces the probabilistic reasoning problems to solving linear programs and standard satisfiability tests with respect to the underlying description logic. Later the extension was adapted to $\mathcal{SHIF}(D)$ and $\mathcal{SHOIN}(D)$ in [5]. The property of reducing the probabilistic reasoning problems allows for a modular implementation reusing mature software components, i.e., a DL reasoner and a linear program solver. Although this approach seems attractive, experience shows that, usually, severe performance problems can be expected for expressive logics. However, since the P-$\mathcal{SHOQ}(D)$ approach seems well-suited for a modular implementation we investigate the hypothesis that such a implementation can be done in practice. As a DL reasoner RacerPro has been chosen due to its stability and maturity [6, 7]. For solving linear programs we used the mosek [8] and the or-objects[9] solver. The combined probabilistic reasoning system is called ContraBovemRufum[1]. The main contributions of this paper can be summarised as follows:

1. We investigated the feasibility of the approach taken by Giugno and Lukasiewicz [1] in a practical implementation, showing that it can be implemented. The implementation already includes the optimised algorithms proposed in [5].
2. Some techniques for further optimisations of the reasoning algorithms are discussed.
3. Modeling experiements demonstrate some of the difficulties that come with the semantics of this kind of probabilistic description logics.

* This paper has been partially funded by the Project PreSInt DFG NE 279/9-1

[1] See [10] for the genesis of the name.

2 Syntax and Semantics

We use $\mathcal{SHIQ}(D)$ as underling DL in our system, hence the investigated probabilistic description logic is named P-$\mathcal{SHIQ}$(D). For syntax and semantics of $\mathcal{SHIQ}(D)$ the reader is referred to [11–13]. The probabilistic part is introduced in the remainder of this section.

In order to extend a description logic for dealing with probabilistic knowledge an additional syntactical and semantical construct is needed. The additional construct is called a *conditional constraint*.

A conditional constraint consists of a statement of conditional probability for two concepts C, D as well as a lower bound l and an upper bound u on that probability. It is written as follows: $(D|C)[l, u]$ Where C can be called the *evidence* and D the *hypothesis*.

To gain the ability to store such statements in a knowledge base it has to be extended to a probabilistic knowledge base $\mathcal{PKB}$. Additionally to the TBox $\mathcal{T}$ of a description logic knowledge base we introduce the PTBox $\mathcal{PT}$, which consists of $\mathcal{T}$ and a set of conditional constraints $\mathcal{D}_g$, and a PABox $\mathcal{PA}$ holding sets of conditional constraints $\mathcal{D}_o$ for each probabilistic individual o. We also define the set of probabilistic individuals I_p, which contains all individuals o for which some probabilistic knowledge is available and therefore a set $\mathcal{D}_o$. In [1] there is no ABox declared, knowledge about so called classical individuals is also stored inside the TBox using nominals. $\mathcal{D}_g$ therefore represents statistical knowledge about concepts and $\mathcal{D}_o$ represents degrees of belief about the individual o.

To be able to define the semantics for a description logic with probabilistic extension the interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot)$ has to be extended by a probability function μ on the domain of interpretation $\Delta^{\mathcal{I}}$. The extended interpretation is called the *probabilistic interpretation* $\mathcal{P}r = (\mathcal{I}, \mu)$. Each individual o in the domain $\Delta^{\mathcal{I}}$ is mapped by the probability function μ to a value in the interval [0,1] and the values of all $\mu(o)$ have to sum up to 1 for any probabilistic interpretation $\mathcal{P}r$. With the probabilistic interpretation $\mathcal{P}r$ at hand the probability of a concept C, represented by $\mathcal{P}r(C)$, is defined as sum of all $\mu(o)$ where $o \in C^{\mathcal{I}}$. The probabilistic interpretation of a conditional probability $\mathcal{P}r(D|C)$ is given as $\frac{\mathcal{P}r(C \sqcap D)}{\mathcal{P}r(C)}$ where $\mathcal{P}r(C) > 0$.

A conditional constraint $(D|C)[l, u]$ is *satisfied* by $\mathcal{P}r$ or $\mathcal{P}r$ *models* $(D|C)[l, u]$ if and only if $\mathcal{P}r(D|C) \in [l, u]$. We will write this as $\mathcal{P}r \models (D|C)[l, u]$. A probabilistic interpretation $\mathcal{P}r$ is said to *satisfy* or *model* a terminology axiom T, written $\mathcal{P}r \models T$, if and only if $\mathcal{I} \models T$. A set $\mathcal{F}$ consisting of terminological axioms and conditional constraints, where F denotes the elements of $\mathcal{F}$, is satisfied or modeled by $\mathcal{P}r$ if and only if $\mathcal{P}r \models F$ for all $F \in \mathcal{F}$.

The *verification* of a conditional constraint $(D|C)[l, u]$ is defined as $\mathcal{P}r(C) = 1$ and $\mathcal{P}r$ has to be a model of $(D|C)[l, u]$. We also may say $\mathcal{P}r$ *verifies* the conditional constraint $(D|C)[l, u]$. On the contrary the *falsification* of a conditional constraint $(D|C)[l, u]$ is given if and only if $\mathcal{P}r(C) = 1$ and $\mathcal{P}r$ does **not** satisfy $(D|C)[l, u]$. It is also said that $\mathcal{P}r$ *falsifies* $(D|C)[l, u]$.

Further a conditional constraint F is said to be *tolerated* under a terminology $\mathcal{T}$ and a set of conditional constraints $\mathcal{D}$ if and only if a probabilistic interpretation $\mathcal{P}r$ can be found that verifies F and $\mathcal{P}r \models \mathcal{T} \cup \mathcal{D}$.

With all these definitions at hand we are now prepared to define the *z-partition* of a set of generic conditional constraints $\mathcal{D}_g$. The z-partition is build as ordered partition $(\mathcal{D}_0, \ldots, \mathcal{D}_k)$ of $\mathcal{D}_g$, where each part $\mathcal{D}_i$ with $i \in \{0, \ldots, k\}$ is the set of all conditional constraints $F \in \mathcal{D}_g \setminus (\mathcal{D}_0 \cup \cdots \cup \mathcal{D}_{i-1})$, that are tolerated under the generic terminology $\mathcal{T}_g$ and $\mathcal{D}_g \setminus (\mathcal{D}_0 \cup \cdots \cup \mathcal{D}_{i-1})$.

If the z-partition can be build from a PABox $\mathcal{PT} = (\mathcal{T}, \mathcal{D}_g)$, it is said to be *generically consistent* or *g-consistent*. A probabilistic knowledge base $\mathcal{PKB} = (\mathcal{PT}, (\mathcal{P}_o)_{o \in I_p})$ is *consistent* if and only if $\mathcal{PT}$ is g-consistent and $\mathcal{P}r \models \mathcal{T} \cup \mathcal{D}_o$ for all $o \in I_p$.

We use the z-partition for the definition of the lexicographic order on the probabilistic interpretations $\mathcal{P}r$ as follows:

A probabilistic interpretation $\mathcal{P}r$ is called *lexicographical preferred* to a probabilistic interpretation $\mathcal{P}r'$ if and only if some $i \in \{0, \ldots, k\}$ can be found, that $|\{F \in \mathcal{D}_i \mid \mathcal{P}r \models F\}| > |\{F \in \mathcal{D}_i \mid \mathcal{P}r' \models F\}|$ and $|\{F \in \mathcal{D}_j \mid \mathcal{P}r \models F\}| = |\{F \in \mathcal{D}_j \mid \mathcal{P}r' \models F\}|$ for all $i < j \leq k$.

We say a probabilistic interpretation $\mathcal{P}r$ of a set $\mathcal{F}$ of terminological axioms and conditional constraints is a *lexicographically minimal model* of $\mathcal{F}$ if and only if no probabilistic interpretation $\mathcal{P}r'$ is lexicographical preferred to $\mathcal{P}r$.

By now the meaning of *lexicographic entailment* for conditional constraints from a set $\mathcal{F}$ of terminological axioms and conditional constraints under a PTBox $\mathcal{PT}$ is given as:

A conditional constraint $(D|C)[l, u]$ is a *lexicographic consequence* of a set $\mathcal{F}$ of terminological axioms and conditional constraints under a PTBox $\mathcal{PT}$, written as $\mathcal{F} \mid\!\sim (D|C)[l, u]$ under $\mathcal{PT}$, if and only if $\mathcal{P}r(D) \in [l, u]$ for every lexicographically minimal model $\mathcal{P}r$ of $\mathcal{F} \cup \{(C|\top)[1, 1]\}$. *Tight lexicographic consequence* of $\mathcal{F}$ under $\mathcal{PT}$ is defined as $\mathcal{F} \mid\!\sim_{tight} (D|C)[l, u]$ if and only if l is the infimum and u is the supremum of $\mathcal{P}r(D)$. We define $l = 1$ and $u = 0$ if **no** such probabilistic interpretation $\mathcal{P}r$ exists.

Finally we define lexicographic entailment using a probabilistic knowledge base $\mathcal{PKB}$ for generic and assertional conditional constraints F.

If F is a generic conditional constraint, then it is said to be a lexicographic consequence of $\mathcal{PKB}$, written $\mathcal{PKB} \mid\!\sim F$ if and only if $\emptyset \mid\!\sim F$ under $\mathcal{PT}$ and a tight lexicographic consequence of $\mathcal{PKB}$, written $\mathcal{PKB} \mid\!\sim_{tight} F$ if and only if $\emptyset \mid\!\sim_{tight} F$ under $\mathcal{PT}$.

If F is an assertional conditional constraint for $o \in I_P$, then it is said to be a lexicographic consequence of $\mathcal{PKB}$, written $\mathcal{PKB} \mid\!\sim F$, if and only if $\mathcal{D}_o \mid\!\sim F$ under $\mathcal{PT}$ and a tight lexicographic consequence of $\mathcal{PKB}$, written $\mathcal{PKB} \mid\!\sim_{tight} F$ if and only if $\mathcal{D}_o \mid\!\sim_{tight} F$ under $\mathcal{PT}$.

3 ContraBovemRufum System

The ContraBovemRufum system implements the algorithms presented in [1, 5] in the Java programming language. Figure 1 displays the generic system architecture. For the reasoning tasks RacerPro with its JRacer interface is used. Therefore it supports the probabilistic description logic P-$\mathcal{SHIQ}$(D). As solvers a native Java solver by Opsresearch and the Mosek linear program solver have been integrated.

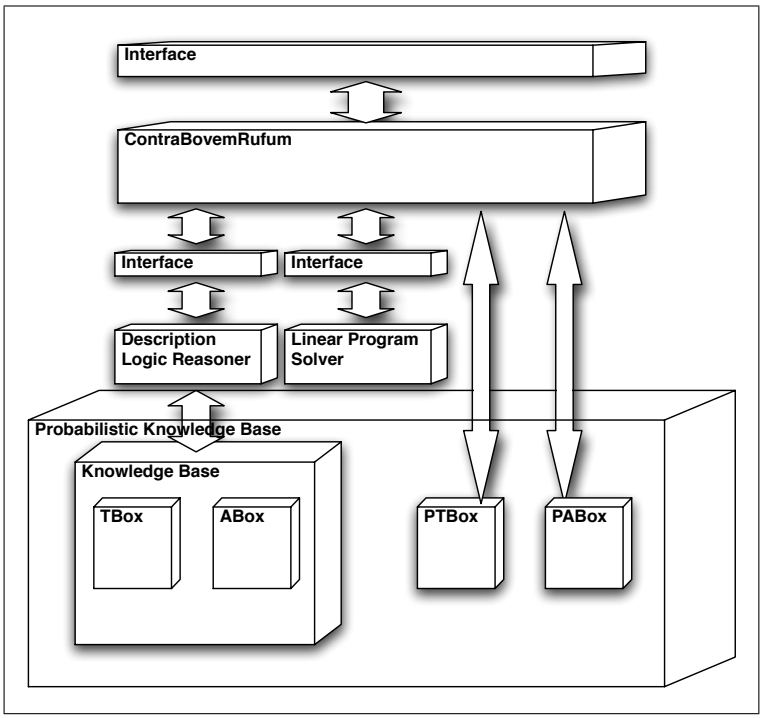

Fig. 1. System architecture

For application programmers two different sets of interfaces are provided. The first set contains the ProbabilisticKBInterface, which provides all operations related to setting up and modifying PTBox and PABox, and the ProbabilisticEntailmentInterface, which offers the probabilistic inference operations to decide consistency for $\mathcal{PT}$ and $\mathcal{PKB}$ as well as probabilistic subsumption and probabilistic instance checking.

The second set of interfaces handles the configuration of the services. Using the SolverConfiguration interface the selection of the solver may be changed at runtime. The ReasonerConfiguration interface makes the settings for the rea-

soner. With the EntailmentConfiguration interface the algorithm used to compute the entailment may be chosen at runtime. Currently tight logical entailment and the two available algorithms for tight lexicographic entailment are supported.

The motivation for the following discussion is to present implementation issues on the one hand, and point to possibilities for optimisation techniques on the other. As previously mentioned the proposed approach for solving probabilistic reasoning problems relies on use of standard reasoning services provided by a description logic reasoner in order to build linear programs which may be handed over to a solver in order to decide the solvability of the programs.

In order to decide satisfiability the first objective is to build a set $\mathcal{R}_{\mathcal{T}}(\mathcal{F})$. It contains the elements r, which map conditional constraints $F_i = (D_i|C_i)[l_i, u_i]$, elements of a set of conditional constraints $\mathcal{F}$, onto one of the following terms $D_i \sqcap C_i$, $\neg D_i \sqcap C_i$ or $\neg C_i$ under the condition, that the intersection of our r is not equal to the bottom concept given the terminology $\mathcal{T}$, written $\mathcal{T} \not\models r(F_1) \sqcap \cdots \sqcap r(F_n) \sqsubseteq \bot$. In the following we will write $\sqcap r$ instead of $r(F_1) \sqcap \cdots \sqcap r(F_n)$ as an abbreviation.

An already implemented idea was to determine the elements r in a tree structure. First the intersection of r of the first two conditional constraints are built. Then these are tested against the terminology $\mathcal{T}$ for satisfiablility. The ones that do satisfy are then intersected with the next mapped conditional constraint and so on. Further extending on this idea one may compute these in parallel. This would require a DL reasoner which is able to handle multiple satisfiability tests at once. Such a behaviour could also be achieved if multiple instances of a DL reasoner are run.

With the set $\mathcal{R}_{\mathcal{T}}(\mathcal{F})$ at hand we are able to set up linear programs to decide the satisfiability of the terminology $\mathcal{T}$ and a finite set of conditional constraints $\mathcal{F}$. The constraints of the linear program are displayed in Figure 2. We say that $\mathcal{T} \cup \mathcal{F}$ is satisfiable if and only if the linear program with the constraints 1a-d is solvable for variables y_r, where $r \in \mathcal{R}_{\mathcal{T}}(\mathcal{F})$. This means that in the objective function all coefficients preceding the variables y_r are set to 1. We further need to introduce the meaning of $r \models C$ which is used as index of the summation in 1a and 1b. It is an abbreviation for $\emptyset \models \sqcap r \sqsubseteq C$. So the coefficient preceding the variables y_r is set in linear constraints 1a and 1b if either $r \models \neg D \sqcap C$ or $r \models D \sqcap C$ may be proven.

Why is the creation of linear programs reasonable? Consider the following: By definition a conditional constraint is satisfied if $u \geq \mathcal{P}r(D|C) \geq l \Leftrightarrow u\mathcal{P}r(C) \geq \mathcal{P}r(D \sqcap C) \geq l\mathcal{P}r(C)$. This may lead us to linear constraints 1a and 1b. Lets focus on the upper bound, whose derivation is displayed in Figure 3. The derivation for the lower bound 1a follows analogously. The linear constraints 1c and 1d reflect that all $\mu(o)$ have to sum up to 1 and all $\mu(o) \in [0, 1]$

Lets have a look at the number of subsumption tests which need to be performed to set up the system of linear constraints. At a first glance, finding a $\models$ under each sum, one might say four tests per variable and conditional constraint. Looking closer we discover that only two are required because they are identical

$$\sum_{r\in\mathcal{R}_\mathcal{T}(\mathcal{F}),r\models\neg D\sqcap C} -ly_r + \sum_{r\in\mathcal{R}_\mathcal{T}(\mathcal{F}),r\models D\sqcap C} (1-l)y_r \geq 0 \quad \text{(for all } (D|C)[l,u]\in\mathcal{F}\text{)} \tag{1a}$$

$$\sum_{r\in\mathcal{R}_\mathcal{T}(\mathcal{F}),r\models\neg D\sqcap C} uy_r + \sum_{r\in\mathcal{R}_\mathcal{T}(\mathcal{F}),r\models D\sqcap C} (u-1)y_r \geq 0 \quad \text{(for all } (D|C)[l,u]\in\mathcal{F}\text{)} \tag{1b}$$

$$\sum_{r\in\mathcal{R}_\mathcal{T}(\mathcal{F})} y_r = 1 \tag{1c}$$

$$y_r \geq 0 \quad \text{(for all } r\in\mathcal{R}_\mathcal{T}(\mathcal{F})\text{)} \tag{1d}$$

Fig. 2. Constraints of the linear program

$$u \sum_{r\in\mathcal{R}_\mathcal{T}(\mathcal{F}),r\models C} y_r \geq \sum_{r\in\mathcal{R}_\mathcal{T}(\mathcal{F}),r\models D\sqcap C} y_r \Leftrightarrow \tag{2a}$$

$$u \sum_{r\in\mathcal{R}_\mathcal{T}(\mathcal{F}),r\models(\neg D\sqcap C)\sqcup(D\sqcap C)} y_r \geq \sum_{r\in\mathcal{R}_\mathcal{T}(\mathcal{F}),r\models D\sqcap C} y_r \Leftrightarrow \tag{2b}$$

$$u \sum_{r\in\mathcal{R}_\mathcal{T}(\mathcal{F}),r\models\neg D\sqcap C} y_r + u \sum_{r\in\mathcal{R}_\mathcal{T}(\mathcal{F}),r\models D\sqcap C} y_r \geq \sum_{r\in\mathcal{R}_\mathcal{T}(\mathcal{F}),r\models D\sqcap C} y_r \Leftrightarrow \tag{2c}$$

$$\sum_{r\in\mathcal{R}_\mathcal{T}(\mathcal{F}),r\models\neg D\sqcap C} uy_r + \sum_{r\in\mathcal{R}_\mathcal{T}(\mathcal{F}),r\models D\sqcap C} (u-1)y_r \geq 0 \tag{2d}$$

Fig. 3. Upper bound derivation

for lower and upper bound. But even this may be optimised further. Considering that the $\sqcap r$ represents a map of all conditional constraints on $D_i \sqcap C_i$, $\neg D_i \sqcap C_i$ or $\neg C_i$ and they are tested on subsumbtion against $D \sqcap C$ and $\neg D \sqcap C$ we observe that only if the first subsumbtion test of $\sqcap r \sqsubseteq D \sqcap C$ failed the second one is necessary. Therefore significantly reducing the number of required tests per variable and conditional constraint.

With the tool at hand to decide satisfiability, we may also decide if a conditional constraint may be tolerated by a set of conditional constraints $\mathcal{F}$. To verify a constraint we add a conditional constraint $(C|\top)[1,1]$. With the extended set the linear program is generated and solved. If an unfeasible solution is computed the conditional constraint is conflicting. If an optimal solution is found, the conditional constraint is tolerated. Now the z-partition of a set of conditional constraints is computable. An idea for optimisation at this point is to compute $\mathcal{R}_\mathcal{T}(\mathcal{D}_g)$ once and only append the concept C to $\sqcap r$ for each verified constraint in order to obtain the needed $\mathcal{R}_\mathcal{T}(\mathcal{F})$. This would significantly reduce the number of satisfiability tests which need to be done.

How to compute tight probability bounds for given evidence C and conclusion D in respect to a set of conditional constraints $\mathcal{F}$ under a terminology $\mathcal{T}$? The task is named *tight logical entailment* and denoted $\mathcal{T} \cup \mathcal{F} \models_{tight} (D|C)[l,u]$.

Given that $\mathcal{T} \cup \mathcal{F}$ is satisfiable, a linear program is set up for $\mathcal{F} \cup (C|\top)[1,1] \cup (D|\top)[0,1]$. The objective function is set to $\sum_{r \in R, r \models D} y_r$. So the coefficient in front of the variables y_r are set 1 if $r \models D$. The tight logical entailed lower bound l is computed by minimising, respectively the upper bound u by maximising the linear program.

In order to compute tight probabilistic lexicographic entailment for given evidence C and conclusion D under a $\mathcal{PKB}$ the following steps have to be taken:

1. Compute the z-partition of $\mathcal{D}_g$ in order to be able to generate a lexicographic ordering
2. Compute lexicographic minimal sets $\mathcal{D}'$ of conditional constraints of $\mathcal{D}_g$ as elements of $\overline{\mathcal{D}}$.
3. Compute the tight logical entailment $\mathcal{T} \cup \mathcal{F} \cup \mathcal{D}' \models_{tight} (D|C)[l,u]$ for all $\mathcal{D}' \in \overline{\mathcal{D}}$.
4. Select the minimum of all computed lower bounds and the maximum of all upper bounds.

The 2. step needs some explanation since a new task "compute *lexicographic minimal sets*" is introduced. In order to define a lexicographic minimal set $\mathcal{D}'$, a preparatory definition is required. A set $\mathcal{D}' \subset \mathcal{D}_g$ *lexicographic preferable* to $\mathcal{D}'' \subset \mathcal{D}_g$ if and only if some $i \in \{0, \ldots, k\}$ exists such that $|\mathcal{D}' \cap \mathcal{D}_i| > |\mathcal{D}'' \cap \mathcal{D}_i|$ and $|\mathcal{D}' \cap \mathcal{D}_i| > |\mathcal{D}'' \cap \mathcal{D}_i|$ for all $i < j \leq k$. With the lexicographic order introduced onto the sets $\mathcal{D}'$ the definition of lexicographic minimal is given as: $\mathcal{D}'$ is lexicographic minimal in $\mathcal{S} \subseteq \{S|S \subseteq \mathcal{D}_g\}$ if and only if $\mathcal{D}' \in \mathcal{S}$ and no $\mathcal{D}'' \in \mathcal{S}$ is lexicographic preferable to $\mathcal{D}'$.

4 Modeling with Conditional Constraints

In order to understand the semantics that comes with the new system some simple modeling experiments are presented here. Our initial grasp was that the conditional constraints can be used to model a degree of overlap between two concepts as depicted in figure 4.

In the first experiment we start out by simply adding one conditional constraint with crisp bounds to the PTBox with respect to an empty TBox. Therefore no restrictions come from the TBox. For example to model that some concept A overlaps with some other concept X to a degree of 60% (see Figure 4) $(X|A)[0.6, 0.6]$ is placed into the PTbox.

With PTBox containing only this one constraint we were interested in the entailed tight lexicographic bounds for $(A|X)[?,?]$. From intuition one would expect the answer to be $(A|X)[0,1]$. Also if we have a look at $Pr(A|X) = \frac{Pr(A \sqcap X)}{Pr(X)}$ with $Pr(X) = 1$ from verifying X in order to compute the entailment and $Pr(A \sqcap X) = Pr(X|A)Pr(A) = 0.6Pr(A)$ from the given conditional constraint therefore $Pr(A)$ remains unknown which should lead to an answer of ignorance $[0,1]$. But our systems answer is $(A|X)[0.0, 0.0]$. This says that concepts A and X should be mutually exclusive. Which is somehow contradicting with our intention

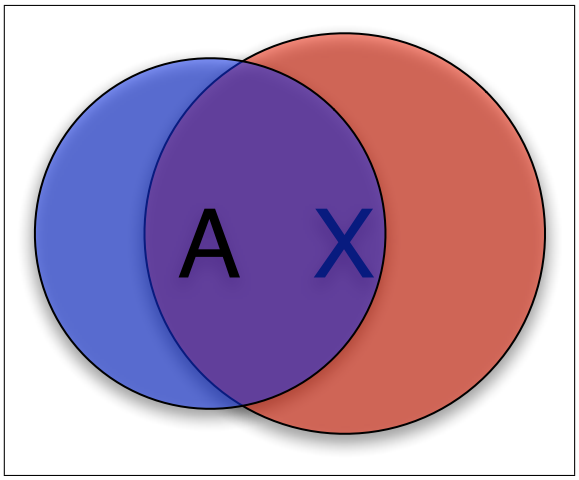

Fig. 4. A overlap X

which we wanted to specify with $(X|A)[0.6, 0.6]$ in the first place. Investigating the Linear Program (see Figure 5) which is solved last one can see that 0.0 is obviously the only solution.

$$
\begin{array}{rrrrrr}
y_{X\sqcap A}+ & y_{\neg X\sqcap A} & & & = & max \\
0.4y_{X\sqcap A}- & 0.6y_{\neg X\sqcap A} & & & \geq & 0 \\
-0.4y_{X\sqcap A}+ & 0.6y_{\neg X\sqcap A} & & & \geq & 0 \\
0y_{X\sqcap A}- & y_{\neg X\sqcap A}+ & 0y_{X\sqcap\neg A}- & y_{\neg X\sqcap\neg A} & \geq & 0 \\
0y_{X\sqcap A}+ & y_{\neg X\sqcap A}+ & 0y_{X\sqcap\neg A}+ & y_{\neg X\sqcap\neg A} & \geq & 0 \\
y_{X\sqcap A}+ & y_{\neg X\sqcap A}+ & y_{X\sqcap\neg A}+ & y_{\neg X\sqcap\neg A} & = & 1 \\
 & & & y_r & \geq & 0
\end{array}
$$

Fig. 5. Linear Program

The same result is observed if the crisp bounds of the conditional constraint in the PTBox are relaxed as long as $u \neq 1$. This behavior appears because Pr(A) = 0 is accepted as a possible solution and therefore $(X|A)[0.6, 0.6]$ is determined to be in the lexicographic minimal set. If we would enforce that $Pr(A) > 0$ than the previous constraint is forced out of the lexicographic minimal set and the answer becomes $[0, 1]$.This can for example be achieved by adding a conditional constraint $(A|\top)[0.00001, 1]$ to the PTBox. It is an ad-hoc solution which we choose in order to test the system without modifying it. Also we observed a

numeric problem as the lower bound was pushed closer to zero and the old $[0.0, 0.0]$ answer was returned. In email communication Mr. Lukasiewicz pointed to an elegant solution by maximizing the variables containing A and see if the result is > 0. With this solution the linear program remains the same only changes in the objective function are required.

As we previously have tweaked the system such that it entailed ignorance instead of bottom/empty set with respect to $(A|X)$ we thought it possible to remove the ignorance regarding X overlap with A from the probabilistic knowledge base. For example if we look at our figure 4 again we might want to include additionally the conditional constraint $(A|X)[0.4, 0.4]$ in the PTBox. Modeling this additional knowledge is not possible with the given semantics because both constraints are mutually in conflict with the other. The system can not decide which of the two conditional constraints should be ranked higher. Therefore the z-Partion can not be build and the knowledge base becomes inconsistent.

The presented experiments show that we either must search for a new intuition to describe how one should model with the system or adapt a different system which captures better the intuition previously described.

5 Related Systems

In 2005, Thomas Lukasiewicz made the nmproblog system[14] available as an compiled executable. The system computes entailment in variable strength nonmonotonic probabilistic logics (System P, System Z, lexicographic) on top of propositional logic. Having introduced the ContraBovemRufum System on top of Racer in [10] recently the Pronto System[15] has been announced on top of the Pellet Reasoner and is now also available. These systems are based on expressive description logics. Further the ContraBovemRufum System has been integrated with the ontology matching system HMatch 2.0[16] in order to facilitate mapping validation[17].

6 Conclusion

We presented the ContraBovemRufum system which provides probabilistic lexicographic reasoning services with respect to a probabilistic knowledge base and discussed its underlying techniques. Thus proving that a modular implementation can be achieved. The mentioned optimization ideas suggest that there is some potential to increase the systems performance.

As demonstrated in section 4 modeling with the system is not an easy task. Already with a simple example we were able to demonstrate some of the very tricky parts of the provided semantics. If we wish to use the system for modeling probabilistic knowledge an intuition needs to be developed that characterizes the semantics of a conditional constraint better than the idea of a degree of overlap between two concepts.

If the reader is interested in verifying the presented results on his own, the implementation can be obtained from the following site under the section software:
`http://www.sts.tu-harburg.de/%7Et.naeth/`

References

1. Giugno, R., Lukasiewicz, T.: P-$\mathcal{SHOQ}(\mathbf{D})$: A probabilistic extension of $\mathcal{SHOQ}(\mathbf{D})$ for probabilistic ontologies in the semantic web. In: JELIA. (2002) 86–97
2. Jaeger, M.: Probabilistic reasoning in terminological logics. In: KR. (1994) 305–316
3. Heinsohn, J.: Probabilistic description logics. In de Mantaras, R.L., Poole, D., eds.: Proc. of the 10th Conf. on Uncertainty in Artificial Intelligence, Seattle, Washington (1994) 311–318
4. Koller, D., Levy, A.Y., Pfeffer, A.: P-classic: A tractable probabilistic description logic. In: AAAI/IAAI. (1997) 390–397
5. Lukasiewicz, T.: Expressive probabilistic description logics. Artif. Intell. **172**(6-7) (2008) 852–883
6. Racer Systems GmbH & Co. KG: Racerpro user's guide (2005) Version 1.9.
7. Racer Systems GmbH & Co. KG: Racerpro reference manual (2005) Version 1.9.
8. MOSEK ApS: The mosek java api manual version 4.0. MOSEK ApS (2006)
9. DRA Systems: Or-objects (2000)
10. Näth, T.H.: Analysis of the average-case behavior of an inference algorithm for probabilistic description logics. Diplomarbeit, TU Hamburg-Harburg (February 2007)
11. Haarslev, V., Möller, R.: Expressive abox reasoning with number restrictions, role hierarchies, and transitively closed roles. In Giunchiglia, F., Selman, B., eds.: Proceedings of Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000), Breckenridge, Colorado, USA, 12-15 April, Morgan Kaufmann (2000) 273–284
12. Horrocks, I., Sattler, U., Tobies, S.: Reasoning with individuals for the description logic SHIQ. In MacAllester, D., ed.: Proceedings of the 17th International Conference on Automated Deduction (CADE-17), Germany, Springer Verlag (2000)
13. Haarslev, V., Möller, R.: RACER System Description. Volume 2083. (2001) 701–705 Available at http://www.racer-systems.com.
14. Lukasiewicz, T.: Nonmonotonic probabilistic logics under variable-strength inheritance with overriding: Algorithms and implementation in nmproblog. In: 4th International Symposium on Imprecise Probabilities and Their Applications, Pittsburgh, Pennsylvania. (2005)
15. Clark & Parsia, LLC.: Pronto (03 2008)
16. Castano, S., Ferrara, A., Montanelli, S.: Matching ontologies in open networked systems: Techniques and applications. Journal on Data Semantics (JoDS) **V** (2006)
17. Castano, S., Ferrara, A., Lorusso, D., Näth, T.H., Möller., R.: Mapping validation by probabilistic reasoning. In: To appear in: 5th European Semantic Web Conference (ESWC 2008). (2008)