Reducing the Size of the Optimization Problems in Fuzzy Ontology Reasoning

Fernando Bobillo and Umberto Straccia

1 Dpt. of Computer Science & Systems Engineering, University of Zaragoza, Spain
2 Istituto di Scienza e Tecnologie dell’Informazione (ISTI - CNR), Pisa, Italy
Email: fbobillo@unizar.es, straccia@isti.cnr.it

Abstract. Fuzzy ontologies allow the representation of imprecise structured knowledge, typical in many real-world application domains. A key factor in the practical success of fuzzy ontologies is the availability of highly optimized reasoners. This short paper discusses a novel optimization technique: a reduction of the size of the optimization problems obtained during the inference by the fuzzy ontology reasoner fuzzyDL.

1 Introduction

In recent years, we have noticed an increase in the number of applications for mobile devices that could benefit from the use of semantic reasoning services [1]. Because of the limited capabilities of mobile devices, it is especially important to develop reasoning algorithms performing efficiently in practice. In order to deal with imprecise knowledge, such applications could use fuzzy ontologies [8]. In fuzzy ontologies, concepts and relations are fuzzy. Consequently, the axioms are not in general either true or false, but they may hold to some degree of truth.

However, little effort has been paid so far to the study and implementation of optimization techniques for fuzzy ontology reasoning, which is essential to reason with real-world scenarios in practice (some exceptions are [3,4,5,6]). This short paper discusses some optimization techniques to improve the performance of the reasoning algorithm by reducing the size of optimization problems obtained during the inference. In particular, we will provide optimized MILP encodings of the restrictions involving n-ary operators and fuzzy membership functions. Such optimizations have been implemented in fuzzyDL, arguably the most popular and advanced fuzzy ontology reasoner [2], and proved their usefulness.

2 Background on fuzzyDL reasoning

We assume the reader to be familiar with the syntax and semantics of fuzzy Description Logics (DLs) [8]. The reasoning algorithm implemented in fuzzyDL combines tableaux rules with an optimization problem. After some preprocessing, fuzzyDL applies tableau rules decomposing complex concept expressions into simpler ones, as usual in tableau algorithms, but also generating a system of inequation constraints. These inequations have to hold in order to respect
the semantics of the DL constructors. After all rules have been applied, an optimization problem must be solved before obtaining the final solution. The tableau rules are deterministic and the optimization problem is unique.

This optimization problem has a solution iff the fuzzy KB is consistent. In fuzzyDL, we obtain a bounded Mixed Integer Linear Programming (MILP) problem, that is, minimising a linear function with respect to a set of constraints that are linear inequations in which rational and integer variables can occur. The problem is bounded, with rational variables ranging over $[0, 1]$ and some integer variables ranging over $\{0, 1\}$. For example, in Lukasiewicz fuzzy DLs, the restriction $x_1 \otimes_L x_2 = z$ can be encoded using the set of constraints $\{x_1 + x_2 - 1 \leq z, x_1 + x_2 - 1 \geq z - y, z \leq 1 - y, y \in \{0, 1\}\}$. Observe that the MILP encoding of the restriction has introduced a new variable $y$: the two possibilities $y = 0$ and $y = 1$ encode the non-deterministic choice implicit in the interpretation of the conjunction under Lukasiewicz fuzzy logic. The complexity of solving a MILP problem is NP-complete and it depends on the number of variables, so it is convenient to reduce the number of new variables.

Let $x, z$ be $[0, 1]$-variables, and $x_u$ be a rational unbounded variable. fuzzyDL has to solve some restrictions involving fuzzy connectives, such as $x_1 = \ominus x_2$, $x_1 \otimes x_2 = z$, $x_1 \oplus x_2 = z$, or $x_1 \Rightarrow x_2 = z$. Furthermore, it also needs to solve some restrictions $d(x_u) \geq z$ involving fuzzy membership functions $d$ such as the trapezoidal $(k_1, k_2, q_1, q_2, q_3, q_4)$ (see Table 1 (a)), the triangular $(k_1, k_2, q_1, q_2, q_3)$, left $(k_1, k_2, q_1, q_2)$, or right $(k_1, k_2, q_1, q_2)$.

### 3 Optimizing Lukasiewicz N-ary Operators

Let us start with the case of conjunction concepts in Lukasiewicz fuzzy DLs. An $n$-ary concept of the form $(C_1 \cap C_2 \cap \cdots \cap C_n)$ can be represented, using associativity, only using binary conjunctions $(C_1 \cap (C_2 \cap (\cdots \cap C_n) \cdots))$. A binary conjunction concept introduces a restriction of the form $x_1 \otimes_L x_2 = z$ which, as shown in Section 2, can be encoded adding a new binary variable $y$. Hence, in order to represent the $n$-ary conjunction, $n - 1$ new variables $y_i$ would be needed. However, it is possible to give a more efficient representation by considering the conjunction as an $n$-ary operator. Indeed, a restriction of the form $x_1 \otimes_L x_2 \otimes \cdots \otimes x_n = z$ can be encoded using only one new binary variable and, thus, saves $2^n - 2$ possible alternative assignments to the variables $y_i$.

$$\sum_{i=1}^{n} x_i - (n - 1) \leq z,$$

$$y \leq 1 - z,$$

$$\sum_{i=1}^{n} x_i - (n - 1) \geq z - (n - 1)y,$$

$$y \in \{0, 1\}.$$


\begin{table}
\begin{center}
\begin{tabular}{|c|c|c|c|}
\hline
i & y_3 & y_2 & y_1 & \text{condition} \\
\hline
1 & 0 & 0 & 0 & x_1 = z \\
2 & 0 & 0 & 1 & x_2 = z \\
3 & 0 & 1 & 0 & x_3 = z \\
4 & 0 & 1 & 1 & x_4 = z \\
5 & 1 & 0 & 0 & x_5 = z \\
\hline
\end{tabular}

\end{center}
\end{table}

Table 1. (a) Trapezoidal membership function; (b) Encoding of 5 states.

\[ y = 0 \] encodes the case \[ z = \sum_{i=1}^{n} x_i - (n-1) \geq 0, \] and \[ y = 1 \] encodes the case \[ z = 0 \] and \[ \sum_{i=1}^{n} x_i - (n-1) < 0. \] Let us consider now disjunction concepts in Lukasiewicz fuzzy DLs. A binary disjunction can be represented adding a new binary variable \( y \) as \( \{ x_1 + x_2 \leq z + y, y \leq z, x_1 + x_2 \geq z, y \in \{0,1\} \} \). Again, \( n-1 \) new binary variables would be needed but, similarly as before, considering the disjunction as an \( n \)-ary operator we would need only one new binary variable:

\[
\sum_{i=1}^{n} x_i \leq z + (n-1)y,
\]

\[ y \leq z, \]

\[
\sum_{i=1}^{n} x_i \geq z,
\]

\[ y \in \{0,1\}. \]

4 Optimizing Gödel N-ary Operators

An \( n \)-ary conjunction can be represented using binary conjunctions adding restrictions of the form \( x_1 \otimes_G x_2 = z \), which can be encoded as follows:

\[
z \leq x_1,
\]

\[
z \leq x_2,
\]

\[
x_1 \leq z + y,
\]

\[
x_2 \leq z + (1 - y),
\]

\[ y \in \{0,1\}. \]

The idea is that if \( y = 0, x_1 = z \) is the minimum; whereas if \( y = 1, x_2 = z \) is the minimum. This adds a new variable \( y \), so in the case of \( n \)-ary conjunctions there would be \( n-1 \) new variables. Treating the conjunction as an \( n \)-ary operator, a more efficient representation is possible. An \( n \)-ary conjunction introduces a restriction of the form \( x_1 \otimes_G x_2 \otimes \cdots \otimes x_n = z \). To represent that the minimum of \( n \) variables \( x_i \) is equal to \( z \), we can use \( n \) binary variables \( y_i \) such that if \( y_i \)
takes the value 0 then \( x_i \) (representing the minimum) is equal to \( z \), and such that the sum of the \( y_i \) is 1, so \( z \) takes the value of some \( x_i \). Note that the minimum may not be unique. Such a representation is as follows:

\[
\begin{align*}
z &\leq x_i, \text{ for } i \in \{1, \ldots, n\}, \\
x_i &\leq z + y_i, \text{ for } i \in \{1, \ldots, n\}, \\
\sum_{i=1}^{n} y_i &\leq 1, \\
y_i &\in \{0, 1\}, \text{ for } i \in \{1, \ldots, n\}.
\end{align*}
\]

Now, we will show that it is possible to give a more efficient representation, Essentially, we need to encode \( n \) possible states. However, \( n \) possible states can be encoded using \( m = \lceil \log_2 n \rceil \) new binary variables only. For instance, for \( n = 5 \), only \( \lceil \log_2 5 \rceil = 3 \) binary variables are necessary, where we use the encoding of the \( n = 5 \) states in Table 1(b).

The main point is now to correctly encode the condition \( x_i \leq z + y_i \) of the old encoding. We proceed as follows. Let \( b_i \) be a string of length \( m \), representing the value \( i - 1 \) in base 2 (\( 1 \leq i \leq n \)). For instance, for \( i = 4 \), \( b = 011 \), as illustrated in the table above. Let us define the expression \( e_{ij} \) (\( 1 \leq i \leq n, 1 \leq j \leq m \)) as:

\[
e_{ij} = \begin{cases} 
y_j & \text{if the } j \text{th bit of } b_i \text{ is 0} \\
1 - y_j & \text{otherwise.}
\end{cases}
\]

For \( i = 4 \), we have \( b = 011 \) and, thus, \( e_{41} = 1 - y_1, e_{42} = 1 - y_2, \) and \( e_{43} = y_3 \). Now we are ready to provide the whole encoding:

\[
\begin{align*}
z &\leq x_i, \text{ for } i = 1, \ldots, n \\
x_i &\leq z + \sum_{j=1}^{m} e_{ij}, \text{ for } i = 1, \ldots, n \\
\sum_{j=1}^{m} 2^{j-1} y_j &\leq n - 1, \\
y_j &\in \{0, 1\}, \text{ for } j = 1, \ldots, m.
\end{align*}
\]

The first condition is the same as before. The second condition guarantees that \( x_i \leq z \) in the state \( b_i \). Finally, the third condition ensures that we are not addressing more than \( n \) states. For instance, for \( n = 5 \) we have:

\[
\begin{align*}
z &\leq x_1, \\
z &\leq x_2, \\
z &\leq x_3, \\
z &\leq x_4, \\
z &\leq x_5, \\
x_1 &\leq z + y_1 + y_2 + y_3.
\end{align*}
\]
\[ x_2 \leq z + (1 - y_1) + y_2 + y_3, \]
\[ x_3 \leq z + y_1 + (1 - y_2) + y_3, \]
\[ x_4 \leq z + (1 - y_1) + (1 - y_2) + y_3, \]
\[ x_5 \leq z + y_1 + y_2 + (1 - y_3), \]
\[ y_1 + 2y_2 + 4y_3 \leq 4, \]
\[ y_1 \in \{0, 1\}, \]
\[ y_2 \in \{0, 1\}, \]
\[ y_3 \in \{0, 1\}. \]

The case of the disjunction in Gödel fuzzy DLs is dual. If an \( n \)-ary concept of the form \((C_1 \sqcup C_2 \sqcup \cdots \sqcup C_n)\) is represented using binary disjunctions, \( n - 1 \) new binary variables are needed. However, if we consider it as an \( n \)-ary concept, it is possible to use \( \lceil \log_2 n \rceil \) new binary variables only:

\[
\begin{align*}
  &z \geq x_i, \text{ for } i = 1, \ldots, n \\
  &x_i + \sum_{j=1}^{m} e_{ij} \geq z \text{ for } i = 1, \ldots, n \\
  &\sum_{j=1}^{m} 2^{i-1} y_j \leq n - 1, \\
  &y_j \in \{0, 1\} \text{ for } j = 1, \ldots, m.
\end{align*}
\]

5 Optimizing Fuzzy Membership Functions

Let us start with the case of trapezoidal functions, which introduce a restriction of the form \( \text{trapezoidal}(k_1, k_2, q_1, q_2, q_3, q_4)(x_u) \geq z \). A restriction of that form can be represented by adding 5 new binary variables \( y_i \) as follows:

\[
\begin{align*}
  &x_u + (k_1 - q_1)y_2 \geq k_1, \\
  &x_u + (k_1 - q_2)y_3 \geq k_1, \\
  &x_u + (k_1 - q_3)y_4 \geq k_1, \\
  &x_u + (k_1 - q_4)y_5 \geq k_1, \\
  &x_u + (k_2 - q_1)y_1 \leq k_2, \\
  &x_u + (k_2 - q_2)y_3 \leq k_2, \\
  &x_u + (k_2 - q_3)y_5 \leq k_2, \\
  &x_u + (k_2 - q_4)y_4 \leq k_2, \\
  &x_u \leq 1 - y_1 - y_5, \\
  &x_u \geq y_3, \\
  &x_u + (q_1 - q_2)x_u + (k_2 - q_1)y_2 \leq k_2, \\
  &x_u + (q_1 - q_2)x_u + (k_1 - q_2)y_2 \geq k_1 + q_1 - q_2.
\end{align*}
\]
\[
x_u + (q_4 - q_3)x_u + (k_2 - q_3)y_4 \leq k_2 + q_4 - q_3,\\
x_u + (q_4 - q_3)x_u + (k_1 - q_4)y_4 \geq k_1,\\
y_1 + y_2 + y_3 + y_4 + y_5 = 1,\\
y_i \in \{0, 1\}, \text{ for } i = 1, \ldots, 5.
\]

To reduce now the number of binary variables, the idea is to have 5 binary variables encoding the 5 possible states: \(x_u \leq q_1 \) \((y_1 = 1)\), \(x_u \in [q_1, q_2] \) \((y_2 = 1)\), \(x_u \in [q_2, q_3] \) \((y_3 = 1)\), \(x_u \in [q_3, q_4] \) \((y_4 = 1)\), and \(x_u \geq q_4 \) \((y_5 = 1)\). However, as shown in Table [2](b), it is possible to represent 5 states using only 3 variables.

The case of other fuzzy membership functions is similar. In triangular functions, a naive encoding introduces 4 new variables to represent the 4 possible states, but it is possible to consider only 2. Finally, in left and right shoulder functions, it is necessary to consider 3 states, which can be achieved by adding 2 new binary variables, instead of the 3 ones needed in the non-optimal encoding.

By considering the fact that even for moderate sized ontologies we may easily generate thousands of such constraints, it is evident that the number of saved binary variables \(n\), and hence the number of saved assignments \(2^n\), is non-negligible.

**Acknowledgement** This research work has been partially supported by the CICYT project TIN2013-46238-C4-4-R and DGA-FSE.

**References**