Language equations for approximate matching in the Description Logic $\mathcal{FL}_0$

Franz Baader and Pavlos Marantidis*
firstname.lastname@tu-dresden.de

Theoretical Computer Science, TU Dresden, Germany

Abstract

Both matching and unification in the Description Logic $\mathcal{FL}_0$ can be reduced to solving certain formal language equations. In previous work, we have extended unification in $\mathcal{FL}_0$ to approximate unification, and have shown that approximate unification can be reduced to approximately solving language equations. An approximate solution of a language equation need not make the languages on the left- and right-hand side of the equation equal, but close w.r.t. a given distance function. In the present paper, we consider approximate matching. We show that, for a large class of distance functions, approximate matching is in NP. We then consider a particular distance function $d_1(K, L) = 2^{-n}$, where n is the length of the shortest word in the symmetric difference of the languages K, L, and show that w.r.t. this distance function approximate matching is polynomial.

1 Introduction and previous work

Matching is the special case of unification where one of the sides to be unified has no variables and thus remains unchanged under substitutions. In Description Logics (DLs), matching concepts against patterns (concepts with variables) was introduced to help filter out unimportant aspects of complicated concepts appearing in large industrial knowledge bases [4]. Unification of patterns was suggested as a means to detect redundancies in ontologies, by finding different concepts that may potentially stand for the same intuitive notion [3]. For the DL $\mathcal{FL}_0$, unification and matching have been investigated in detail in [3]. From an equational theory point of view, this is unification and matching modulo the equational theory ACUIh of a binary associative, commutative, and idempotent function symbol with a unit and several homomorphisms. It was shown in [3] that both problems can be reduced to solving certain formal language equations. In particular, matching can be reduced to formal language equations of the following form: given finite languages (sets of words) $V_0, U_0, U_1, \ldots, U_n$ we want to know whether there exist finite languages $X_1, \ldots, X_n$ such that

$$U_0 = V_0 \cup V_1 \cdot X_1 \cup \cdots \cup V_n \cdot X_n \tag{1}$$

where "$\cdot$" stands for concatenation of languages. A *solution* of such an equation is an *assignment* σ of languages to the variables X_i such that the above equation holds as equality of languages. An assignment is called *finite* if all the languages $\sigma(X_i)$ are finite. A finite assignment that is a solution is called a *matcher*. In other words, matching in $\mathcal{FL}_0$ reduces to checking whether equations of the form (1) have a matcher. Baader and Narendran [3] showed that this problem is decidable in polynomial time by proving that (1) has a matcher iff the assignment

$$\theta(X_i) := \bigcap_{v \in V_i} v^{-1} U_0 \quad (i = 1, \ldots, n)$$

*Supported by DFG Graduiertenkolleg 1763 (QuantLA).

is a solution of (1). It is easy to see that computing θ and checking whether it actually is a solution can be done in polynomial time.

As an example, consider the equation

$$\{a, ab, abb\} = \{a, ab\} \cdot X.$$

Following the procedure above, for $n = 1$, $U_0 = \{a, ab, abb\}$, $V_0 = \emptyset$ and $V_1 = \{a, ab\}$ we have that $a^{-1}U_0 = \{\varepsilon, b, bb\}$, $(ab)^{-1}U_0 = \{\varepsilon, b\}$ and thus $\theta(X) := \{\varepsilon, b, bb\} \cap \{\varepsilon, b\} = \{\varepsilon, b\}$. Immediately, it can be verified that $\{a, ab\} \cdot \theta(X) = \{a, ab, abb\}$ holds as equality, and thus θ is a solution.

On the other hand, for the equation

$$\{ab\} = \{a, ab\} \cdot X$$

working as before we obtain $a^{-1}U_0 = \{b\}$, $(ab)^{-1}U_0 = \{\varepsilon\}$ and thus $\theta(X) := \{b\} \cap \{\varepsilon\} = \emptyset$. Since $\{a, ab\} \cdot \theta(X) \neq \{ab\}$, this problem does not have a solution.

In [1], matching in extensions of $\mathcal{FL}_0$ by the bottom concept, atomic negation, and number restrictions was considered. The problem again reduces to solving language equations of a form similar to (1), but now using a restricted form of infinite languages. In this setting, one can again construct a candidate solution similar to θ above, but now checking whether this assignment is indeed a solution becomes more involved. For this purpose, the authors of [1] introduce so-called *tree-like automata*, which can be used to realize this test in polynomial time.

Recently, approximate unification was introduced in order to increase the recall of classical unification [2], i.e., find substitutions that are "almost" unifiers rather than exact unifiers. Approximate unification in $\mathcal{FL}_0$ can again be reduced to solving formal language equations, but now approximately. An approximate solution of such an equation does not make the left- and right-hand sides of the equation equal, but instead close w.r.t. a distance function on languages.

In [2], a *language distance* is defined to be a metric on the set of languages over the given alphabet Σ, i.e., a function $d : 2^{\Sigma^*} \times 2^{\Sigma^*} \to [0, \infty)$ satisfying the properties

$(M1)$ $d(K, L) = 0 \iff K = L$

$(M2)$ $d(K, L) = d(L, K)$

$(M3)$ $d(K, M) \leq d(K, L) + d(L, M)$.

A common approach to define a language distance function is to "measure" the size of the symmetric difference of the input languages [6, 5, 2], i.e., define $d(K, L) := m(K \triangle L)$ where $K \triangle L := (K \setminus L) \cup (L \setminus K)$ and m is an appropriate function. This way, $(M2)$ is guaranteed to hold. If m is actually a measure (in the mathematical sense), $(M3)$ holds as well, (since $K \triangle M \subseteq K \triangle L \cup L \triangle M$). $(M1)$ corresponds to the requirement that $m(L) = 0$ iff $L = \emptyset$.

An interesting example of such a language distance introduced in the literature [2, 5, 6] is the function

$$d_1(K, L) := 2^{-n},$$

where $n = \min\{\, |w| \,:\, w \in K \triangle L \,\}$. The intuition underlying this distance is that differences between the two languages are less important if they occur for longer words. The function considers the length n of the shortest word in the symmetric difference of the input languages and yields 2^{-n} as distance, which becomes smaller if n gets larger. Approximate unification in $\mathcal{FL}_0$ w.r.t. d_1 (or rather, w.r.t. the concept distance induced by this language distance) was shown to be of the same complexity as exact unification, i.e., ExpTime-complete [2].

2 Approximate matching

In the present paper, we consider approximate matching rather than approximate unification in $\mathcal{FL}_0$. To this purpose, we need to solve language equations of the form (1) approximately. We will first show that this problem is in NP for a wide class of distances, and then prove that, for d_1, we can even get a polynomial-time algorithm.

Definition 1 (Approximate language matching)**.** *Given an equation of the form (1), the* approximate language matching problem *w.r.t. the language distance d with threshold $p \in \mathbb{Q}$ asks whether there exists a finite assignment σ such that*

$$d(U_0, V_0 \cup V_1 \cdot \sigma(X_1) \cup \cdots \cup V_n \cdot \sigma(X_n)) < p.$$

Such an assignment, if one exists, is called an approximate matcher*. A (not necessarily finite) assignment that satisfies the above inequality is called an* approximate solution*.*

w.r.t. d_1 with threshold 2^{-2} we have that the assignment $\sigma_1(X) = \{b\}$ is an approximate matcher while $\sigma_2(X) = \emptyset$ is not, since

$$d_1(\{ab\}, \{a, ab\} \cdot \sigma_1(X)) = d_1(\{ab\}, \{ab, abb\}) = 2^{-3} < 2^{-2} \text{ while}$$
$$d_1(\{ab\}, \{a, ab\} \cdot \sigma_2(X)) = d_1(\{ab\}, \emptyset) = 2^{-2} \not< 2^{-2}.$$

Furthermore, any assignment σ with $\{b\} \subseteq \sigma(X) \subseteq \{a, b\}^* \setminus \{\varepsilon, a\}$ is an approximate solution for the same threshold.

Since we want to show complexity results, the *size* of a problem has to be defined formally. As usual, we use $|w|$ to denote the length of a word w. Note however that we will use the same notation for denoting the cardinality of a set S. The *size* $\|L\|$ of a language L is the sum of the lengths of all its words. The size of an approximate language matching problem is the sum of the sizes of the languages $U_0, V_0, V_1, \ldots, V_n$ plus the number of bits required for representing the threshold p. The size of an assignment σ, with $\sigma(X_i) = L_i$, is defined to be the sum of the sizes of the languages $L_1, \ldots, L_n$.

Following standard notation in the literature of formal languages, we will omit "$\cdot$" when we refer to concatenation. Furthermore, the left quotient of $L \subseteq \Sigma^*$ w.r.t. $v \in \Sigma^*$ is defined to be $v^{-1}L = \{w \in \Sigma^* \mid vw \in L\}$. The set of words over Σ of length $\bowtie m$ is $\Sigma^{\bowtie m} = \{w \in \Sigma^* \mid |w| \bowtie m\}$, where $\bowtie \in \{\leq, \geq\}$.

In this section, we prove that, for language distances that satisfy the property

$$K \triangle L \subseteq M \triangle N \implies d(K, L) \leq d(M, N), \tag{2}$$

the existence of an approximate solution implies the existence of an approximate matcher of polynomial size. If the distance is computable in polynomial time, this yields an NP-algorithm for deciding the approximate matching problem. Property (2) holds for all language distance functions of the form $d(K, L) = m(K \triangle L)$ where m is a measure.

Assume that we are given an equation of the form (1), a distance d satisfying property (2), and a threshold p. Let σ be an approximate solution, with $\sigma(X_i) = M_i$ for every $i = 1, \ldots, n$. The following two observations suffice to prove our result:

1. Let $m = \max\{|w| : w \in U_0\}$. Then σ' with $\sigma'(X_i) = \sigma(X_i) \cap \Sigma^{\leq m}$ is also an approximate matcher. Obviously, introducing words that are longer than the longest word in U_0 only adds words to the right hand side of the equation (1), and thus to the symmetric difference. By (2), removing such words cannot increase the distance value. Consequently, if there is an approximate solution, then there also is one whose words have length at most $m \leq \|U_0\|$.

2. Every word $w \in \sigma(X_i)$ introduces a set of words $S_w := V_i\{w\}$ on the right-hand side. There are two possibilities: either $S_w \cap (U_0 \setminus V_0) \neq \emptyset$ or $S_w \subseteq V_0 \cup (\Sigma^* \setminus U_0)$. In the second case, we can simply omit w from $\sigma(X_i)$ and get the same or lower distance between the left- and the right-hand side. As for the first case, there are at most $|Suf(U_0)|$ many such words w, where $Suf(L) = \{u \in \Sigma^* : \exists w \in L. \exists v \in \Sigma^*. vu = w\}$. Note that $|Suf(L)| \leq \|L\| + |L|$, which is linearly bounded by the size of L.

Lemma 2. *Assume that d is a language distance function satisfying property (2). An approximate matching problem w.r.t. d has an approximate solution iff it has an approximate matcher of size at most quadratic in the size of the problem.*

By guessing an assignment of polynomial size and then checking in polynomial time whether it actually is an approximate matcher for the given threshold value, we obtain an NP-algorithm for approximate matching.

Theorem 3. *Given an equation of the form (1), a threshold p, a distance d satisfying property (2), the approximate matching problem is decidable in NP.*

3 Approximate Matching w.r.t. d_1

For the language distance d_1, we can actually get a better complexity result: the approximate matching problem is decidable in polynomial time.

Since d_1 is monotone w.r.t. the symmetric difference of the input languages, it suffices to check whether there is an approximate solution. In the positive case, Lemma 2 then guarantees the existence of an approximate matcher of polynomial size. Looking at the definition of d_1, it is easy to see that, if the input languages agree on all words of length up to $m-1$, their distance is at most 2^{-m}. More generally, we have the following:

Lemma 4. *Let K, L be languages over Σ and $p \leq 2^{-m}$, $m \in \mathbb{N}$. Then,*

$$d_1(K,L) < p \iff K \cap \Sigma^{\leq m} = L \cap \Sigma^{\leq m} \iff K \cup \Sigma^{\geq m+1} = L \cup \Sigma^{\geq m+1}.$$

As an easy consequence of this lemma, we obtain:

Proposition 5. *Given an equation of the form (1), the assignment $\sigma(X_i) = L_i$ is an approximate solution w.r.t. d_1 with threshold $p \leq 2^{-m+1}$ iff $U_0 \cup \Sigma^{\geq m} = V_0 \cup V_1 L_1 \cup \cdots \cup V_n L_n \cup \Sigma^{\geq m}$, i.e., iff it is a solution of the equation*

$$U_0 \cup \Sigma^{\geq m} = V_0 \cup V_1 X_1 \cup \cdots \cup V_n X_n \cup \Sigma^{\geq m}. \tag{3}$$

Reflecting on the equation $\{ab\} = \{a, ab\}X$ w.r.t. d_1 with threshold $p = 2^{-2}$ we have that

$$\{ab\} \cup \Sigma^{\geq 3} = \{ab, abb\} \cup \Sigma^{\geq 3} = \{a, ab\}\,\sigma_1(X) \cup \Sigma^{\geq 3},$$

thus verifying again that σ_1 is an approximate solution. In fact, for any assignment σ with $\{b\} \subseteq \sigma(X) \subseteq \{a,b\}^* \setminus \{\varepsilon, a\}$ it holds that

$$\{ab\} \cup \Sigma^{\geq 3} = \{a, ab\}\,\sigma_1(X) \cup \Sigma^{\geq 3}.$$

Meanwhile,

$$\{ab\} \cup \Sigma^{\geq 3} \neq \emptyset \cup \Sigma^{\geq 3} = \{a, ab\}\,\sigma_2(X) \cup \Sigma^{\geq 3},$$

since σ_2 is not an approximate solution.

By Proposition 5, finding an approximate solution w.r.t. d_1 with $p \leq 2^{-m+1}$ for the equation of the form (1) reduces to finding a solution for the equation (3). Adapting the technique used in [3] for matching in $\mathcal{FL}_0$, we get the following result.

Lemma 6. *An equation of the form (3) has a solution iff*

$$\sigma(X_i) = L_i := \bigcap_{v \in V_i} \left(v^{-1}(U_0 \cup \Sigma^{\geq m})\right)$$

is a solution.

Proof. The if direction is trivial.

For the only-if direction, assume that $\tau(X_i) = M_i$ is a solution of (3). We want to show that

$$U_0 \cup \Sigma^{\geq m} = V_0 \cup V_1 L_1 \cup \cdots \cup V_n L_n \cup \Sigma^{\geq m}. \tag{4}$$

Obviously, the equation holds for all words of length at least m. Suppose that $w \in U_0$ and $|w| < m$. Since τ is a solution, either $w \in V_0$ or $w \in V_i M_i$ for some $i \in \{1, \ldots, n\}$. In the first case, there is nothing more to show. In the second case, there are words $v_0 \in V_i, u \in M_i$ s.t. $w = v_0 u$. Since τ is a solution, for every word $v \in V_i$ it holds that $vu \in U_0 \cup \Sigma^{\geq m}$. Thus $u \in \bigcap_{v \in V_i} v^{-1}(U_0 \cup \Sigma^{\geq m}) = L_i$, which proves language inclusion in one direction.

For the other direction, since there is a solution, it holds that $V_0 \subseteq U_0 \cup \Sigma^{\geq m}$. Thus it suffices to prove that $V_i L_i \subseteq U_0 \cup \Sigma^{\geq m}$. Assume that $w \in V_i L_i$. This means that $w = v_0 \ell$ for some $v_0 \in V_i$ and $\ell \in L_i$. Thus we get

$$\ell \in \bigcap_{v \in V_i} v^{-1}(U_0 \cup \Sigma^{\geq m}) \implies \ell \in {v_0}^{-1}(U_0 \cup \Sigma^{\geq m}) \implies v_0 \ell \in U_0 \cup \Sigma^{\geq m},$$

which completes the proof. □

Applying the above lemma to our running example, we obtain

$$\begin{aligned} \sigma(X) :&= a^{-1}(\{ab\} \cup \{a,b\}^{\geq 3}) \cap (ab)^{-1}(\{ab\} \cup \{a,b\}^{\geq 3}) \\ &= (\{b\} \cup \{a,b\}^{\geq 2}) \cap (\{\varepsilon\} \cup \{a,b\}^{\geq 1}) \\ &= (\{b\} \cup \{a,b\}^{\geq 2}) = \{a,b\}^* \setminus \{\varepsilon, a\}, \end{aligned}$$

which we have already seen that is an approximate solution.

Checking in polynomial time whether the assignment from Lemma 6 is actually a solution can be done by using tree-like automata (see [3] for details). Overall, we obtain the following for d_1.

Theorem 7. *Given an equation of the form (1), the approximate matching problem w.r.t. d_1 with threshold p is decidable in polynomial time.*

4 Future work

We will investigate whether the NP-upper bound stated in Theorem 3 is sharp, i.e., whether there are language distances in the introduced class for which the approximate matching problem is NP-hard. We are also interested in finding non-trivial language distances other than d_1 for which approximate matching is polynomial. As mentioned above, matching in the extensions of $\mathcal{FL}_0$ investigated in [1] reduces to solving language equations that involve infinite languages. Hence, we would like to investigate whether the results of this paper can be extended to such equations.

References

[1] Baader, F., Küsters, R., Borgida, A., McGuinness, D.L.: Matching in description logics. J. of Logic and Computation 9(3), 411–447 (1999)

[2] Baader, F., Marantidis, P., Okhotin, A.: Approximate unification in the description logic $\mathcal{FL}_0$. In: JELIA-16. Lecture Notes in Computer Science, vol. 10021, pp. 49–63. Springer (2016)

[3] Baader, F., Narendran, P.: Unification of concept terms in description logics. J. of Symbolic Computation 31(3), 277–305 (2001)

[4] Borgida, A., McGuinness, D.L.: Asking queries about frames. In: Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96). pp. 340–349 (1996)

[5] Kephart, D.E.: Topology, morphisms, and randomness in the space of formal languages. Ph.D. thesis, University of South Florida (2005)

[6] Vianu, V.: The Bodnarchuk metric space of languages and the topology of the learning space. In: Mathematical Foundations of Computer Science 1977, Proceedings. pp. 537–542 (1977)