Applied Mathematics Letters 25 (2012) 1394–1398

Contents lists available at SciVerse ScienceDirect

Applied Mathematics Letters

journal homepage: www.elsevier.com/locate/aml

Implication of regular expressions

Alex Thomo *

Department of Computer Science, University of Victoria, PO Box 3055, STN CSC, University of Victoria, BC V8W 3P6, Canada

ARTICLE INFO

Article history:
Received 9 September 2010
Received in revised form 1 December 2011
Accepted 5 December 2011

Keywords:
Regular expressions
Incomplete information
Complexity

ABSTRACT

In this work we study the following implication problem for regular expressions: "Given a set of regular expressions **R** and a regular expression S, is it true that every string which matches the regular expressions in **R** also matches S?" The problem comes in two flavors: "non-disjoint" and "disjoint". We show that both of them are PSPACE-complete. While the complexity for the first variant is not surprising – the problem is coNP-complete even for very simple patterns (given by wildcards) – the complexity result for the second variant represents a big jump since the problem is in PTIME for the case of patterns with wildcards. Towards the goal of charting the boundary of tractability, we then present an analysis of when the problem remains in PTIME.

1. Introduction

Imagine a database of strings which are not (always) available or too expensive to be accessed and suppose that we have incomplete information on these strings. The incomplete information on a string is represented by a set of patterns given by regular expressions that the string matches. Our task here is to decide whether such a set of patterns implies the presence of a new pattern. This is called "pattern implication" in strings.

The importance of this problem is also evident from a data versus expression complexity point of view (cf. [1,2]). If we consider the string to be "data" and the patterns to be "query" (or "view") definitions, then determining that a pattern is "implied" tells us with certainty that a query is satisfiable without accessing the data, i.e. without incurring "data-complexity".

There are two semantics with respect to this problem. In the first, the substrings matching the patterns can overlap. In the second, they are disjoint. Therefore, we have two notions of pattern implication: "non-disjoint" and "disjoint."

Libkin and Sirangelo consider in [3] the implication of patterns with wildcards. They show that for this class of patterns, the non-disjoint implication is coNP-complete, whereas the disjoint implication is in PTIME if the number of leading and trailing wildcards is fixed.

Here we show that for patterns given by (full-fledged) regular expressions both implication problems become PSPACE-complete. While the complexity of non-disjoint implication is not surprising, there is a big jump in the complexity of disjoint implication. Therefore, it is interesting to know more about the boundary of tractability for this problem.

Towards this goal, we present a large class ($\mathcal{C}$) of consequents (patterns that we want to imply) for which the problem has only one source of complexity: the structure of the consequent itself. Specifically, if a pattern S that we want to imply is such that a DFA for $\Sigma^* S \Sigma^*$ can be built in polynomial time, then the disjoint implication can be decided in polynomial time as well, no matter how complex the antecedent regular expressions are.

The rest of the work is organized as follows. In Section 2 we give definitions and characterizations. In Section 3 we present our complexity results for the general case. In Section 4 we characterize a special class of consequents. Finally, Section 5 concludes the work.

* Tel.: +1 250 472 5786; fax: +1 250 472-5708.
E-mail address: thomo@cs.uvic.ca.

0893-9659/$ – see front matter
doi:10.1016/j.aml.2011.12.009

2. Definitions and characterizations

Let Σ be a fixed alphabet. A *regular pattern* (*RP*) is a regular expression over Σ. We will denote RPs by $R, S, \ldots$. For ease of notation we will blur the distinction between regular expressions and the regular languages that they define. Let s be a string over Σ and R an RP. We say that:

Definition 1. *s matches R*, denoted by $s \models R$, if there exists a substring s' of s such that $s' \in R$.

Clearly,

Proposition 1. *$s \models R$ iff $s \in \Sigma^* R \Sigma^*$.*

Let $\mathbf{R} = \{R_1, \ldots, R_n\}$ be a set of RPs. We consider two flavors of pattern matching.

Definition 2.

1. *s matches* $\mathbf{R}$, denoted by $s \models \mathbf{R}$, if s matches every pattern in $\mathbf{R}$.
2. *s disjointly matches* $\mathbf{R}$, denoted by $s \models_d \mathbf{R}$, if there exist disjoint substrings $s_1, \ldots, s_n$ of s such that $s_i \in R_i$, for $i \in [1, n]$.

Now, we consider the following pattern implication problems.

Definition 3.

1. We say that $\mathbf{R}$ *implies* an RP S, denoted by $\mathbf{R} \vdash S$, if $s \models S$ whenever $s \models \mathbf{R}$.
2. We say that $\mathbf{R}$ *disjointly implies* RP S, denoted by $\mathbf{R} \vdash_d S$, if $s \models_d S$ whenever $s \models_d \mathbf{R}$.

It is easy to verify the following two propositions.

Proposition 2. *$s \models \mathbf{R}$ iff $s \in \Sigma^* R_1 \Sigma^* \cap \cdots \cap \Sigma^* R_n \Sigma^*$.*

Proposition 3. *$s \models_d \mathbf{R}$ iff $s \in \Sigma^* R_{i_1} \Sigma^* \cdots \Sigma^* R_{i_n} \Sigma^*$ for some permutation $i_1, \ldots, i_n$ of $1, \ldots, n$.*

Based on Proposition 2, we have:

Proposition 4. *$\mathbf{R} \vdash S$ iff $\Sigma^* R_1 \Sigma^* \cap \cdots \cap \Sigma^* R_n \Sigma^* \subseteq \Sigma^* S \Sigma^*$.*

Likewise, based on Proposition 3, we have:

Proposition 5. *$\mathbf{R} \vdash_d S$ iff $\Sigma^* R_{i_1} \Sigma^* \cdots \Sigma^* R_{i_n} \Sigma^* \subseteq \Sigma^* S \Sigma^*$ for each permutation $i_1, \ldots, i_n$ of $1, \ldots, n$.*

Observe that if $\epsilon \in S$, then S is always implied, regardless of $\mathbf{R}$, or the type of implication. This is because in such a case $\Sigma^* S \Sigma^* = \Sigma^*$. Thus, in the rest of the work, we will assume that $\epsilon \notin S$.

Also, observe the if $\epsilon \in R_i$ for some $i \in [1, n]$, then we can safely remove R_i from $\mathbf{R}$ without any effect, for any type of matching or implication. This is because $\Sigma^* R_i \Sigma^* = \Sigma^*$. Thus, in the rest of the work, we will also assume that $\epsilon \notin R_i$ for $i \in [1, n]$.

3. The general case

Now we show the following theorems. We assume that the patterns in $\mathbf{R}$ and pattern S are all given by regular expressions or non-deterministic finite automata (NFAs).

Theorem 1. *The problem of checking, for an RP S and a set of RPs $\mathbf{R}$, whether $\mathbf{R} \vdash S$ is PSPACE-complete.*

Proof. We show that the problem is in PSPACE. By Proposition 5, checking $\mathbf{R} \vdash S$ amounts to checking that

$$\Sigma^* R_1 \Sigma^* \cap \cdots \cap \Sigma^* R_n \Sigma^* \subseteq \Sigma^* S \Sigma^*$$

which is equivalent to checking the emptiness of

$$\Sigma^* R_1 \Sigma^* \cap \cdots \cap \Sigma^* R_n \Sigma^* \cap (\Sigma^* S \Sigma^*)^c.$$

For this, we can construct an alternating finite-state automaton (AFA) A recognizing the above language. The size of A is polynomial in the combined size of the R_i's and S, as measured by the size of regular expressions or NFAs for them. Now, the emptiness of AFAs is in PSPACE (cf. [4]), and thus, our problem is in PSPACE.

To show the PSPACE-hardness, consider the sets of RPs with only one RP in them, for example $\mathbf{R} = \{R\}$. Then the decision problem becomes

$$\Sigma^* R \Sigma^* \subseteq \Sigma^* S \Sigma^*.$$