Information and Computation 242 (2015) 287–305

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Pattern matching with variables: A multivariate complexity analysis [☆]

Henning Fernau, Markus L. Schmid [*]

Fachbereich 4 – Abteilung Informatik, Universität Trier, D-54286 Trier, Germany

ARTICLE INFO

Article history:
Received 11 September 2013
Received in revised form 25 June 2014
Available online 23 March 2015

Keywords:
Parameterised pattern matching
Function matching
NP-completeness
Membership problem for pattern languages
Morphisms

ABSTRACT

A pattern α, i.e., a string that contains variables and terminals, matches a terminal word w if w can be obtained by uniformly substituting the variables of α by terminal words. Deciding whether a given terminal word matches a given pattern is NP-complete and this holds for several natural variants of the problem that result from whether or not variables can be erased, whether or not the patterns are required to be terminal-free or whether or not the mapping of variables to terminal words must be injective. We consider numerous parameters of this problem (i.e., number of variables, length of w, length of the words substituted for variables, number of occurrences per variable, cardinality of the terminal alphabet) and for all possible combinations of the parameters (and variants described above), we answer the question whether or not the problem is still NP-complete if these parameters are bounded by constants.

1. Introduction

In the present work, a detailed complexity analysis of a computationally hard pattern matching problem is provided. The *patterns* considered in this context are strings containing *variables* from $\{x_1, x_2, x_3, \ldots\}$ and *terminal symbols* from a finite alphabet Σ, e.g., $\alpha := x_1\,\mathtt{a}\,x_1\,\mathtt{b}\,x_2\,x_2$ is a pattern, where $\mathtt{a}, \mathtt{b} \in \Sigma$. We say that a word w over Σ *matches* a pattern α if and only if w can be derived from α by uniformly substituting the variables in α by terminal words. The respective pattern matching problem is then to decide for a given pattern and a given word, whether or not the word matches the pattern. For example, the pattern α from above is matched by the word $u := \mathtt{bacaabacabbaba}$, since substituting x_1 and x_2 in α by $\mathtt{baca}$ and $\mathtt{ba}$, respectively, yields u. On the other hand, α is not matched by the word $v := \mathtt{cbcabbcbbccbc}$, since v cannot be obtained by substituting the variables of α by some words.

To the knowledge of the authors, this kind of pattern matching problem first appeared in the literature in 1979 in form of the membership problem for Angluin's *pattern languages* [3,4] (i.e., the set of all words that match a certain pattern) and, independently, it has been studied by Ehrenfeucht and Rozenberg in [10], where they investigate the more general problem of deciding on the existence of a morphism between two given words (which is equivalent to the above pattern matching problem, if the patterns are *terminal-free*, i.e., they only contain variables).

[☆] A preliminary version [11] of this paper was presented at the conference CPM 2013.
[*] Corresponding author.
E-mail addresses: Fernau@uni-trier.de (H. Fernau), MSchmid@uni-trier.de (M.L. Schmid).

http://dx.doi.org/10.1016/j.ic.2015.03.006
0890-5401/© 2015 Elsevier Inc. All rights reserved.

Since their introduction by Angluin, pattern languages have been intensely studied in the learning theory community in the context of inductive inference (see, e.g., Angluin [4], Shinohara [32], Reidenbach [24,25] and, for a survey, Ng and Shinohara [22]) and, furthermore, their language theoretical aspects have been investigated (see, e.g., Angluin [4], Jiang et al. [19], Ohlebusch and Ukkonen [23], Freydenberger and Reidenbach [12], Bremer and Freydenberger [7]). However, a detailed investigation of the complexity of their membership problem, i.e., the above described pattern matching problem, has been somewhat neglected. Some of the early work that is worth mentioning in this regard is by Ibarra et al. [17], who provide a more thorough worst case complexity analysis, and by Shinohara [33], who shows that matching patterns with variables can be done in polynomial time for certain special classes of patterns. Recently, Reidenbach and Schmid [26,27] identify complicated structural parameters of patterns that, if bounded by a constant, allow the corresponding matching problem to be performed in polynomial time (see also Schmid [29]).

In the pattern matching community, independent from Angluin's work, the above described pattern matching problem has been rediscovered by a series of papers. This development starts with [5] in which Baker introduces so-called *parameterised pattern matching*, where a text is not searched for all occurrences of a specific factor, but for all occurrences of factors that satisfy a given pattern with parameters (i.e., variables). In the original version of parameterised pattern matching, the variables in the pattern can only be substituted by single symbols and, furthermore, the substitution must be injective, i.e., different variables cannot be substituted by the same symbol. Amir et al. [1] generalise this problem to *function matching* by dropping the injectivity condition and in [2], Amir and Nor introduce *generalised function matching*, where variables can be substituted by words instead of single symbols and "don't care" symbols can be used in addition to variables. In 2009, Clifford et al. [9] considered generalised function matching as introduced by Amir and Nor, but without "don't care" symbols, which leads to patterns as introduced by Angluin.

In [2], motivations for this kind of pattern matching can be found from such diverse areas as software engineering, image searching, DNA analysis, poetry and music analysis, or author validation. Another motivation arises from the observation that the problem of matching patterns with variables constitutes a special case of the matchtest for *regular expressions with backreferences* (see, e.g., Câmpeanu et al. [8], Schmid [30]), which nowadays are a standard element of most text editors and programming languages (cf. Friedl [14]). Due to its simple definition, the above described pattern matching paradigm also has connections to numerous other areas of theoretical computer science and discrete mathematics, such as (un-)avoidable patterns (cf. Jiang et al. [18]), word equations (cf. Mateescu and Salomaa [21]), the ambiguity of morphisms (cf. Freydenberger et al. [13]), equality sets (cf. Harju and Karhumäki [16]) and database theory (cf. Barceló et al. [6]).

It is a well-known fact that – in its general sense – pattern matching with variables is an NP-complete problem; a result that has been independently reported several times (cf. Angluin [4], Ehrenfeucht and Rozenberg [10], Clifford et al. [9]). However, there are many different versions of the problem, tailored to different aspects and research questions. For example, in Angluin's original definition, variables can only be substituted by non-empty words and Shinohara soon afterwards complemented this definition in [32] by including the empty word as well. This marginal difference, as pointed out by numerous results, can have a substantial impact on learnability and decidability questions of the corresponding classes of *non-erasing* pattern languages on the one hand and *erasing* pattern languages on the other. If we turn from the languages point of view of patterns to the respective pattern matching task, then, at a first glance, this difference whether or not variables can be erased seems negligible. However, in the context of pattern matching, other aspects are relevant, which for pattern languages are only of secondary importance. For example, requiring variables to be substituted in an *injective* way is a natural assumption for most pattern matching tasks and bounding the maximal length of these terminal words by a constant (which would turn pattern languages into finite languages) makes sense for special applications (cf. Baker [5]). Hence, there are many variants of the above described pattern matching problem, each with its individual motivation, and the computational hardness of all these variants cannot directly be concluded from the existing NP-completeness results.

For a systematic investigation, we consider the following natural parameters: the number of different variables in the pattern, the maximal number of occurrences of the same variable in the pattern, the length of the terminal word, the maximum length of the substitution words for variables and the cardinality of the terminal alphabet. Hence, there are 2^5 different combinations of parameters and coupling these with the 2^3 variants of the pattern matching problem with variables results in 256 individual problems. For each of these problems, the question arises whether or not the parameters can be bounded by (preferably small) constants such that the resulting variant of the pattern matching problem is still NP-complete. In this paper, by giving a brief overview of all the existing related results that we are aware of, we show that answers to many of these questions can already be concluded from the literature. Nevertheless, several important cases have not yet been settled and our main contribution is to close all these remaining gaps, such that we obtain an answer for *all* of these 256 questions. In this regard, we provide dichotomy results (in terms of Schaefer [28]) for the pattern matching problem with variables with respect to the above mentioned parameters.

The main motivation for undertaking such a research task is the following. While many NP-complete problems naturally occur in theory and also in practical situations, it is almost never the case that the actual problem that has to be solved is exactly the one for which NP-completeness has been established. It is rather the case that we face a subproblem of an NP-complete problem, tailored to the context in which it is encountered. This is also the reason why NP-completeness results are usually accompanied with statements like "This problem is still NP-complete, even if ... ". As an example, consider the intensive research that is done on proving NP-completeness for graph problems on restricted classes of graphs as planar graphs, graphs with constant degree and so on or the classical results that the satisfiability problem for Boolean formulas is NP-complete even if every clause contains at most 3 literals. Similarly, if some kind of pattern matching problem with