Theoretical Computer Science 600 (2015) 34–48

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Indeterminate strings, prefix arrays & undirected graphs

Manolis Christodoulakis [a], P.J. Ryan [b], W.F. Smyth [b,c,*,1], Shu Wang [d]

[a] Department of Electrical & Computer Engineering, University of Cyprus, PO Box 20537, 1678 Nicosia, Cyprus
[b] Algorithms Research Group, Department of Computing & Software, McMaster University, Hamilton, Ontario, L8S 4K1, Canada
[c] School of Engineering & Information Technology, Murdoch University, 90 South Street, Murdoch, WA 6150, Australia
[d] IBM Toronto Software Lab, 8200 Warden Avenue, Markham, Ontario, L6G 1C7, Canada

ARTICLE INFO

Article history:
Received 1 March 2013
Received in revised form 25 June 2015
Accepted 26 June 2015
Available online 2 July 2015
Communicated by G. Ausiello

Keywords:
Indeterminate string
Regular string
Prefix array
Prefix table
Feasible array
Undirected graph
Minimum alphabet size
Lexicographical order

ABSTRACT

An integer array $\boldsymbol{y} = \boldsymbol{y}[1..n]$ is said to be ***feasible*** if and only if $\boldsymbol{y}[1] = n$ and, for every $i \in 2..n$, $i \leq i + \boldsymbol{y}[i] \leq n + 1$. A string is said to be ***indeterminate*** if and only if at least one of its elements is a subset of cardinality greater than one of a given alphabet Σ; otherwise it is said to be ***regular***. A feasible array $\boldsymbol{y}$ is said to be ***regular*** if and only if it is the prefix array of some regular string. We show using a graph model that every feasible array of integers is a prefix array of some (indeterminate or regular) string, and for regular strings corresponding to $\boldsymbol{y}$, we use the model to provide a lower bound on the alphabet size. We show further that there is a 1–1 correspondence between labelled simple graphs and indeterminate strings, and we show how to determine the minimum alphabet size σ of an indeterminate string $\boldsymbol{x}$ based on its ***associated graph*** $\mathcal{G}_{\boldsymbol{x}}$. Thus, in this sense, indeterminate strings are a more natural object of combinatorial interest than the strings on elements of Σ that have traditionally been studied.

1. Introduction

Pattern matching in strings — that is, locating all the occurrences of a given pattern in a given text — has been studied for at least half a century. A major breakthrough was the realization that preprocessing the pattern would allow the problem to be solved significantly faster. Perhaps the first form of preprocessing was proposed in the seminal paper by Morris & Pratt [24], which computed the ***border array*** of the pattern; that is, an array $\boldsymbol{\beta}$, of the same length as the pattern $\boldsymbol{p}$, such that $\boldsymbol{\beta}[i]$ is the length of the longest proper prefix of $\boldsymbol{p}[1..i]$ that is also a suffix.

In recent years, a generalization of the classical string pattern matching problem has been introduced, where either the pattern or the text, or both, contain *sets* of symbols at each position, as opposed to a single symbol per position in regular strings. These types of sequences are known as ***indeterminate strings*** and were first introduced in a famous paper by Fischer & Paterson [12], then later studied by Abrahamson [1]. In the last ten years or so, much work has been done by Blanchet-Sadri and her associates (for example, [4]) on "strings with holes" — that is, strings on an alphabet Σ augmented by a single letter, a "hole" or "wildcard", that matches all other symbols in Σ. The monograph [3] summarizes much of the pioneering work in this area. For indeterminate strings in their full generality, the third and fourth authors of this paper

* Corresponding author at: Algorithms Research Group, Department of Computing & Software, McMaster University, Hamilton, Ontario, L8S 4K1, Canada.
E-mail addresses: christodoulakis.manolis@ucy.ac.cy (M. Christodoulakis), ryanpj@mcmaster.ca (P.J. Ryan), smyth@mcmaster.ca (W.F. Smyth), wangs@ca.ibm.com (S. Wang).
URL: http://www.cas.mcmaster.ca/cas/research/algorithms.htm (W.F. Smyth).

[1] The work of the third author was supported in part by Grant No. 8180 from the Natural Sciences and Engineering Research Council of Canada.

http://dx.doi.org/10.1016/j.tcs.2015.06.056
0304-3975/© 2015 Elsevier B.V. All rights reserved.

have collaborated in several papers, especially in the contexts of pattern-matching [16–18,29] and extensions to periodicity [27,28].

In the search for a preprocessing approach to speed up the pattern matching problem on indeterminate strings, it soon became clear that the border array is of limited use. For regular strings $\boldsymbol{x}$, the border array has the desirable property that any border of a border of $\boldsymbol{x}$ is also a border of $\boldsymbol{x}$ — thus $\boldsymbol{\beta}$ implicitly specifies every border of every prefix of $\boldsymbol{x}$. For indeterminate $\boldsymbol{x}$, however, due to the nontransitivity of the match operation, this is not true [29,28]. Hence border arrays cannot be used to speed up pattern matching on indeterminate strings. However, it turns out to be possible to make use of another data structure, the ***prefix array*** $\boldsymbol{\pi}$, in which $\boldsymbol{\pi}[i]$ is the length of the longest substring beginning at position i of $\boldsymbol{x}$ that matches a prefix of $\boldsymbol{x}$.

Apparently the first algorithm for computing the prefix array occurred as a routine in the repetitions algorithm of Main & Lorentz [22]; see also [26, pp. 340–347]. A slightly improved algorithm is given in [21, Section 8.4], and two algorithms for computing a "compressed" prefix array are described in [27]. A comprehensive treatment of prefix array construction algorithms can be found in [5]. As noted above, for regular strings the border array and the prefix array are equivalent: it is claimed in [9,10], and demonstrated in detail in [5], that there are $\Theta(n)$-time algorithms to compute one from the other. On the other hand, as shown in [27], for indeterminate strings the prefix array actually allows all borders of every prefix to be specified, while the border array does not [16,19]. Thus the prefix array provides a more compact and more general mechanism for identifying borders, hence for describing periodicity, in indeterminate strings.

Ref. [27] describes an algorithm that computes the prefix array of any indeterminate string. In this paper we consider the "reverse engineering" problem of computing a string corresponding to a given "feasible" array $\boldsymbol{y}$ — that is, any array that could conceivably be a prefix array. The first reverse engineering problem was introduced in [13,14], where a linear-time algorithm was described to compute a lexicographically least string whose border array was a given integer array — or to return the result that no such string exists. There have been many such results published since, corresponding to other data structures and other conditions; for example, [2,11,15]. In [8] a linear-time algorithm is described to compute a lexicographically least regular string $\boldsymbol{x}$ corresponding to a given feasible array $\boldsymbol{y}$, or to return an error if $\boldsymbol{y}$ corresponds to no regular string.

In this paper we solve the more general reverse engineering problem for any feasible array $\boldsymbol{y}$, regardless of whether it corresponds to a regular string or not. Moreover, we establish a remarkable connection between labelled graphs and indeterminate strings. The remainder of the paper is organized as follows. Section 2 provides preliminary information and all the necessary definitions that are used throughout the paper. In Section 3 we prove the surprising result that every feasible array is in fact a prefix array of some string (regular or indeterminate); further, we characterize the minimum alphabet size of a regular string corresponding to a given prefix array in terms of the largest clique in the negative "prefix" graph $\mathcal{P}^-$. We go on to give necessary and sufficient conditions that a given prefix array is regular. Section 4 establishes the duality between strings (whether regular or indeterminate) and labelled undirected graphs; also it provides a characterization of the minimum alphabet size of an indeterminate string $\boldsymbol{x}$ in terms of the number of "independent" maximal cliques in the "associated graph" $\mathcal{G}_{\boldsymbol{x}}$. Section 5 outlines future work.

2. Preliminaries

Traditionally, a string is a sequence of letters taken from some alphabet Σ. Since we discuss "indeterminate strings" in this paper, we begin by generalizing the definition as follows:

Definition 1. A ***string*** with base alphabet Σ is either empty or else a sequence of nonempty subsets of Σ. A 1-element subset of Σ is called a ***regular*** letter; otherwise it is ***indeterminate***. Similarly, a nonempty string consisting only of regular letters is ***regular***, otherwise ***indeterminate***. The empty string $\boldsymbol{\varepsilon}$ is regular.

All alphabets and all strings discussed in this paper are finite. We denote by Σ' the set of all nonempty subsets of Σ, with $\sigma = |\Sigma|$ and $\sigma' = |\Sigma'| = 2^\sigma - 1$. On a given alphabet Σ, there are altogether $(\sigma')^n$ distinct nonempty strings of length n, of which σ^n are regular.

Definition 2. Two elements λ, μ of Σ' are said to ***match*** (written $\lambda \approx \mu$) if they have nonempty intersection. Two strings $\boldsymbol{x}$, $\boldsymbol{y}$ ***match*** ($\boldsymbol{x} \approx \boldsymbol{y}$) if they have the same length and all corresponding letters match.

Thus two regular letters match if and only if they are equal. But note that for indeterminate letters λ, μ, ν, it may be that $\lambda \approx \mu$ and $\lambda \approx \nu$, while $\mu \not\approx \nu$: for example, $\lambda = \{1, 2\}$, $\mu = 1$, $\nu = 2$.

Definition 3. If a string $\boldsymbol{x}$ can be written $\boldsymbol{x} = \boldsymbol{u_1 v}$ and $\boldsymbol{x} = \boldsymbol{w u_2}$ for nonempty strings $\boldsymbol{v}$, $\boldsymbol{w}$, where $\boldsymbol{u_1} \approx \boldsymbol{u_2}$, then $\boldsymbol{x}$ is said to have a ***border*** of length $|\boldsymbol{u_1}| = |\boldsymbol{u_2}|$.

Note that choosing $\boldsymbol{v} = \boldsymbol{w} = \boldsymbol{x}$ yields the empty border $\boldsymbol{\varepsilon}$ of length 0.