Information and Computation 241 (2015) 302–320

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Checking determinism of regular expressions with counting ☆

Haiming Chen *, Ping Lu

State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China

ARTICLE INFO

Article history:
Received 27 June 2013
Received in revised form 21 October 2014
Available online 11 December 2014

Keywords:
DTD
Nondeterminism locating
Regular expressions
Weak and strong determinism
XML Schema

ABSTRACT

In this paper, we first present characterizations of weak determinism for regular expressions with counting, based on which we develop an $O(|\Sigma_E||E|)$ time algorithm to check whether an expression E with counting is weakly deterministic, where Σ_E is the set of distinct symbols in the expression E. Moreover, our algorithm can locate nondeterminism in subexpressions, which is useful for diagnosing purpose. Then we present a characterization of strong determinism for regular expressions with counting, and study the relations between weak determinism and strong determinism. Based on these, we give an $O(|E|)$ time algorithm to check strong determinism, improving the previous upper bound of $O(|E|^3)$ time. Finally, historically there has been another definition of strong determinism for standard regular expressions, which we call restricted strong determinism in this paper. We extend this definition to expressions with counting, establish the relation between the two definitions, and give an $O(|E|)$ time algorithm to check restricted strong determinism for regular expressions with counting.

1. Introduction

Regular expressions have been widely used in many applications. Different applications may require regular expressions with various extensions or restrictions, among them are deterministic regular expressions. For example, Document Type Definition (DTD) and XML Schema, which are the XML schema languages recommended by W3C, require that the content models should be *weakly deterministic regular expressions*. As another example, *strongly deterministic regular expressions* are used in query languages for XML streams [3]. Informally, weak determinism means that, when matching a word against an expression, a symbol can be matched to only one position in the expression without looking ahead, while strong determinism makes an additional restriction that the use of operators should also be unique in the matching with the word. Weakly deterministic regular expressions have been studied in the literature [4–12], also under the name of *one-unambiguous* regular expressions [5,7,9–11]. Strong determinism (or strong one-unambiguity) has also attracted attentions recently [7,8,3,13].

One basic problem is deciding weak or strong determinism of regular expressions. While deciding weak determinism of a standard regular expression E can be solved in $O(|\Sigma_E||E|)$ time [4] where Σ_E is the set of distinct symbols in E, deciding strong determinism of standard regular expressions is more involved and the up-to-date algorithm runs in $O(|E|^3)$ time [3]. Furthermore, it is known that deciding weak or strong determinism is nontrivial for regular expressions with

☆ Parts of the preliminary work of the present paper have been presented as conference abstracts [1] and [2]. Work supported by the National Natural Science Foundation of China under Grant Nos. 61472405, 61070038.

* Corresponding author.
E-mail addresses: chm@ios.ac.cn (H. Chen), luping@ios.ac.cn (P. Lu).

http://dx.doi.org/10.1016/j.ic.2014.12.001
0890-5401/

counting (RE(#)) [7].[1] The latter is extended from standard regular expressions with iterative expressions (i.e., expressions of the form $E^{[m,n]}$), and is used for instance in XML Schema. For deciding weak determinism of regular expressions in RE(#) an $O(|\Sigma_E||E|)$ time method was given [14]. For deciding strong determinism of regular expressions in RE(#) an $O(|E|^3)$ time algorithm was presented [7].

In this paper we study properties of RE(#) and design an $O(|\Sigma_E||E|)$ time algorithm to check whether an expression in RE(#) is weakly deterministic. Moreover, the algorithm can locate nondeterminism locally.

We also present a characterization for an expression E to be strongly deterministic. Recently, Groz et al. [12] gave an $O(|E|)$ time[2] algorithm for checking weak determinism of standard regular expressions and expressions in RE(#). We further show that theoretically, starting from our characterization and combining their techniques, strong determinism can also be checked in $O(|E|)$ time.

In the literature there are two definitions for strongly deterministic expressions, given by Gelade et al. [7] which we follow in this paper, and given by Koch et al. [3] which will be called here restricted strongly deterministic expressions. The two definitions have subtle differences: it is known that, for example, strong deterministic expressions allow different derivations of ε while restricted strongly deterministic expressions forbid this [7]. However, their precise relation is still not known. In this paper we formally make this relation clear.

Contributions. We derive new characterizations of weak determinism for RE(#). There are two main contributions of the new characterizations: (1) It gives rise to an $O(|\Sigma_E||E|)$ time algorithm of checking weak determinism; (2) Our method has the local nondeterminism-locating feature, which is useful for diagnosing purpose. In this paper, we will give an example to show this.

Next we give a new algorithm to determine strong determinism in $O(|E|)$ time. We extend the notion of star normal form [4] to iterative expressions, and show that a weakly deterministic regular expression is strongly deterministic if and only if it is in weakly star normal form. Then we give an $O(|E|)$ time algorithm to check whether a weakly deterministic expression is in weakly star normal form or not. By combining this method and the algorithm in [12], we can check strong determinism in linear time.

We extend the definition of restricted strong determinism to expressions with counting, and establish the relation between strongly deterministic expressions and restricted strongly deterministic expressions. Based on this relation, we show an $O(|E|)$ time algorithm for determining whether an expression is restricted strongly deterministic. The algorithm can be easily adapt to deciding restricted strong determinism for standard regular expressions, improving the complexity for this problem from $O(|E|^3)$ [3] to $O(|E|)$.

Related work. A majority of work considered weak determinism of standard regular expressions. Brüggemann-Klein [4] presented an algorithm for standard regular expressions to check whether an expression is weakly deterministic based on Glushkov automata. By converting expressions into star normal form, the algorithm can check weak determinism in $O(|\Sigma_E||E|)$ time. In [3] an $O(|E|^3)$ time algorithm was given to check restricted strong determinism of standard regular expressions.

For expressions in RE(#), extensions of the Glushkov construction have been studied [7,8,15]. Relation between strongly deterministic expressions and the corresponding Glushkov automata was set up [7], and a strong determinism checking algorithm was given [7], which runs in $O(|E|^3)$ time. Kilpeläinen [14] presented an $O(|\Sigma_E||E|)$ time algorithm to check weak determinism for RE(#).

In [12] Groz et al. gave $O(|E|)$ time algorithms for checking weak determinism of standard regular expressions and expressions in RE(#). Rather than computing the *first* and *followlast* sets on the syntax tree, the algorithm is based on a structural decomposition of the syntax tree [12].

Parts of the preliminary work of the present paper have been presented as conference abstracts [1] and [2]. In [1] we proposed a preliminary diagnosing algorithm for weak determinism of standard regular expressions which is based on testing the original expressions. In [2] we continued the work initiated in [1] and dealt with the different and more challenging problem of checking strong determinism for RE(#). This paper further establishes the relations of weak determinism and two different definitions of strong determinism given in [7] and [3], and shows that each kind of strong determinism can be checked in $O(|E|)$ time.

Organization. The rest of the paper is organized as follows. Section 2 introduces definitions. In Section 3 some properties of regular expressions in RE(#) are studied. In Section 4, we design an algorithm to check weak determinism for regular expressions in RE(#) and show the local nondeterminism-locating feature of our algorithm by an example. In Section 5 we give $O(|E|)$ time algorithms for determining whether an expression is strongly deterministic or restricted strongly deterministic.

[1] The nontrivialness is illustrated by an example in [7]: $(b?a^{[2,3]})^{[2,2]}b$ is weakly deterministic, but $(b?a^{[2,3]})^{[3,3]}b$ is not.

[2] The difference between $O(|\Sigma_E||E|)$ and $O(|E|)$ is that, the latter is irrelevant to $|\Sigma_E|$, which is not considered as a constant when the alphabet of expressions is infinite, but is a constant and can always be omitted when the alphabet is finite.