Information and Computation 206 (2008) 1204–1218

Contents lists available at ScienceDirect

Information and Computation

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

On determinism versus nondeterminism for restarting automata

Hartmut Messerschmidt*, Friedrich Otto

Fachbereich Elektrotechnik/Informatik, Universität Kassel, D-34109 Kassel, Germany

ARTICLE INFO

Article history:
Received 5 June 2007
Revised 18 February 2008
Available online 7 June 2008

ABSTRACT

A restarting automaton processes a given word by executing a sequence of local simplifications until a simple word is obtained that the automaton then accepts. Such a computation is expressed as a sequence of *cycles*. A nondeterministic restarting automaton M is called *correctness preserving*, if, for each cycle $u \vdash_M^c v$, the string v belongs to the characteristic language $L_C(M)$ of M, if the string u does. Our first result states that for each type of restarting automaton $X \in \{R,RW,RWW,RL,RLW,RLWW\}$, if M is a nondeterministic X-automaton that is correctness preserving, then there exists a deterministic X-automaton M_1 such that the characteristic languages $L_C(M_1)$ and $L_C(M)$ coincide. When a restarting automaton M executes a cycle that transforms a string from the language $L_C(M)$ into a string not belonging to $L_C(M)$, then this can be interpreted as an *error* of M. By counting the number of cycles it may take M to detect this error, we obtain a measure for the influence that errors have on computations. Accordingly, this measure is called *error detection distance*. It turns out, however, that an X-automaton with bounded error detection distance is equivalent to a correctness preserving X-automaton, and therewith to a deterministic X-automaton. This means that nondeterminism increases the expressive power of X-automata only in combination with an unbounded error detection distance.

1. Introduction

From a theoretical point of view restarting automata can be seen as a tool that yields a very flexible generalization of analytical grammars. They are analyzers that implement basic as well as enhanced features of analytical grammars. They were introduced in [3] to model the so-called *analysis by reduction* of natural languages. Analysis by reduction in general facilitates the development and testing of categories for syntactic and semantic disambiguation of sentences of natural languages. It is often used (implicitly) for developing formal descriptions of natural languages based on the notion of *dependency* [9,10,16].

Analysis by reduction consists in stepwise simplifications (reductions) of a given sentence, possibly enriched by syntactical and semantical categories, until a correct simple sentence is obtained. Each simplification replaces a small part of the sentence by an even shorter phrase. These reductions are required to meet the so-called *error preserving property*, which states that an incorrect sentence can never be transformed into a correct sentence, and the *correctness preserving property*, which states that a correct sentence cannot be transformed into an incorrect one.

Here, we formalize analysis by reduction by using nondeterministic restarting automata. To each sentence of the language recognized, a restarting automaton M associates all the corresponding derivations through sequences of reduction steps. These reduction steps transform a word that does not belong to the characteristic language $L_C(M)$ of M to sentential forms that do not belong to this language, either, which shows that M has the *error preserving property*. On the other hand, it is only deterministic restarting automata that in general also satisfy the complementary property of being *correctness preserving*,

* Corresponding author.
E-mail addresses: hardy@theory.informatik.uni-kassel.de (H. Messerschmidt), otto@theory.informatik.uni-kassel.de (F. Otto).

0890-5401/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2008.03.021

which states that any cycle of M that starts from a string belonging to the language $L_C(M)$ will again give a string from that language. Accordingly, a nondeterministic restarting automaton is called *correctness preserving* if it satisfies this additional property.

Because of the importance of the correctness preserving property for modelling analysis by reduction, it is quite natural to study nondeterministic restarting automata that have this property. What is the expressive power of correctness preserving (nondeterministic) restarting automata in comparison to deterministic restarting automata on the one hand and to unrestricted nondeterministic restarting automata on the other hand? In fact, we consider this question for various different types of restarting automata that are distinguished by the way in which they move their read/write window across their tape and by the kind of rewrite operations they are allowed to execute. Actually we distinguish between nine such classes of restarting automata, from R- to RLWW-automata (see Section 2 for the definitions).

By presenting corresponding example languages we will see that for nondeterministic RR(W)(W)-automata, the correctness preserving variants are strictly more expressive than the corresponding deterministic variants. On the other hand, for nondeterministic R(W)(W)- and RL(W)(W)-automata, the correctness preserving variants are just as expressive as the corresponding deterministic variants. In fact, we present constructions that transform a nondeterministic correctness preserving restarting automaton of one of these types into a deterministic restarting automaton of the same type that accepts the same input and characteristic languages. This result, which is a generalization of the corresponding result for simple t-RL-automata established in [11], shows that for these types of restarting automata the correctness preserving property severely restricts the power of nondeterminism. Or, put in more positive terms, it stresses the expressive power of deterministic restarting automata.

Intuitively, the correctness preserving property is a rather severe restriction on the way in which nondeterminism can be exploited by nondeterministic restarting automata. This is supported by the results above. Can we relax the correctness preserving property in such a way that some nondeterminism can be used without obtaining the full expressive power of unlimited nondeterminism? Here, we introduce and study such relaxations of the correctness preserving property in the form of the so-called *error detection distance*.

If M is a nondeterministic restarting automaton that is not correctness preserving, then M can execute cycles of the form $u \vdash_M^c v$, where $u \in L_C(M)$ and $v \notin L_C(M)$. This can be interpreted as an *error* of M. If, starting from the restarting configuration $q_0\mathbin{\text{¢}}v\$$, M detects its error and rejects without completing another cycle, then we say that M has *error detection distance* 1. More generally, if M detects its error after executing at most $i-1$ further cycles starting from $q_0\mathbin{\text{¢}}v\$$, then we say that M has error detection distance i. Thus, error detection distance 0 corresponds to the correctness preserving property, that is, having error detection distance $i > 0$ is a less severe restriction for restarting automata than the correctness preserving property. While for an unrestricted nondeterministic restarting automaton M, $L_C(M)$ can be an NP-complete language [4], we will see that the membership problem for the language $L_C(M)$ is solvable in polynomial time, if M has bounded error detection distance. In fact, the degree of the polynomial time bound depends on the value of the error detection distance of M. Thus, a bounded error detection distance does indeed limit the influence of nondeterminism on the expressive power of nondeterministic restarting automata.

This raises the question of whether we obtain hierachies of language classes based on the minimal error detection distance of restarting automata that accept these languages. However, as we will see this is not the case. In fact, for all types of restarting automata, we will show that the corresponding hierarchies consist of only two levels: error detection distance 0 and unbounded error detection distance. This is shown by presenting constructions that, given a restarting automaton M with error detection distance $i > 0$ as input, yield a restarting automaton M' of the same type as M such that M' accepts the same characteristic language as M, but M' has error detection distance 0, that is, M' is correctness preserving. In combination with our results on correctness preserving restarting automata above, this implies that nondeterministic R(W)(W)- or RL(W)(W)-automata of bounded error detection distance are not more expressive than the corresponding deterministic types of restarting automata. Thus, it is the unbounded error detection distance in combination with nondeterminism that makes nondeterministic restarting automata more expressive than the corresponding deterministic variants.

This paper is structured as follows. After restating the basic definitions on restarting automata in Section 2, we study the correctness preserving property in Section 3. In Section 4, we define the error detection distance, and present the announced polynomial-time algorithm for the uniform membership problem for the class of characteristic languages of restarting automata with fixed error detection distance. Then we study the influence of bounded error detection distance on the expressive power of restarting automata (Section 5). The paper closes with a short summary and some remarks about open problems related to our studies.

2. Definitions and notation

Here, we describe in short the type of restarting automaton we will be dealing with. More details on restarting automata in general can be found in [13,14].

A *two-way restarting automaton*, RLWW-automaton for short, is a nondeterministic machine M with a finite-state control, a flexible tape with end markers, and a read/write window of a fixed size. Formally, it is defined as $M = (Q, \Sigma, \Gamma, \text{¢}, \$, q_0, k, \delta)$, where Q is a finite set of states containing the initial state q_0, Γ is a finite tape alphabet that in addition to the input alphabet Σ may also contain a finite number of so-called auxiliary symbols, and ¢ and \$ are the left and right border markers for the