Information and Computation 206 (2008) 1152–1160

Contents lists available at ScienceDirect

Information and Computation

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

Succinct description of regular languages by weak restarting automata

Martin Kutrib* , Jens Reimann

Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany

ARTICLE INFO

Article history:
Received 26 June 2007
Revised 16 February 2008
Available online 5 June 2008

ABSTRACT

The descriptional complexity of restarting automata is investigated. We distinguish the classes of weak restarting automata, i.e., classical restarting automata accepting exactly the regular languages. In order to investigate the descriptional power gained by the additional structural resources of restarting automata, we study the trade-offs in the number of states when changing the representation to classical finite automata. The bounds shown are tight in the exact number of states. Interestingly, for a particular class we can show the tight bounds $2^n + 1$ and 2^n for DFA and NFA conversion, respectively, by a fooling set technique. So, the power gained by the resources given to restarting automata seems to be different compared with nondeterminism.

1. Introduction

Restarting automata have been introduced in [7] in order to model the so-called "analysis by reduction," which is a technique used in linguistics to analyze sentences of natural languages that have a high degree of free word order. The technique consists of stepwise simplification of an extended sentence such that the (in)correctness of the sentence is not affected. A restarting automaton is a finite state device that works on a flexible tape. Attached to the automaton is a read-write (or lookahead) window of fixed size. The automaton works in several cycles. In one cycle it moves the window from left to right along the tape. Depending on the current state and the current content of the window the automaton can continue to scan the input, can rewrite the window content by some shorter string, can accept the input, can halt without accepting, or can restart, i.e., start a new cycle by placing the window back on the left end of the tape and resetting to the initial state.

Different variants of restarting automata have been investigated, mainly from a computational capacity point of view. Several well-known language families are characterized by some variant in a unified framework. For example, characterizations are known for deterministic and nondeterministic context-free languages [9], for Church–Rosser languages [15,16], and for regular languages [14]. Introductions to and surveys of restarting automata are [17,18], which are valuable sources for further results and references.

Our main interest is in the descriptional power gained by the additional structural resources of restarting automata [19]. For some variants it is known that there are savings in the size of description which cannot be bounded by any recursive function when changing from one type of system to another. A good example for such a non-recursive trade-off are restarting automata characterizing the context-free languages, and restarting automata characterizing the Church–Rosser languages. See [6,19] for further examples, and [12] for a survey of non-recursive trade-offs in general.

Here we are particularly interested in weak restarting automata, i.e., restarting automata that accept exactly the regular languages. Since regular languages have many representations in the world of finite automata it is natural to investigate the succinctness of these representations in order to optimize space requirements. Here we consider the number of states as complexity measure. Throughout the paper we always assume deterministic finite automata (DFA) to be complete. It is

* Corresponding author.
E-mail address: kutrib@informatik.uni-giessen.de (M. Kutrib).

0890-5401/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2008.03.016

well known that nondeterministic finite automata (NFA) can offer exponential saving in the number of states compared with deterministic finite automata, but the problem to convert a given DFA to an equivalent minimal NFA is PSPACE-complete [10]. Furthermore, asymptotically tight bounds are $O(n^n)$ for the two-way DFA to one-way DFA conversion, $O(2^{n^2})$ for the two-way NFA to one-way DFA conversion, and 2^n for the one-way NFA to one-way DFA conversion. For finite languages over a k-letter alphabet the NFA to DFA conversion has been solved in [20] with a tight bound of $O(k^{\frac{n}{\log_2 k+1}})$. A valuable source for further results and references is [4].

The paper is organized as follows. The next section contains preliminaries and basics on restarting automata. In order to meet our objective, we have to distinguish the variants of restarting automata that accept exactly the regular languages. We discuss this problem in the next and solve it in the last section. Section 3 is devoted to the trade-offs in the number of states when changing the representation from nondeterministic R(1)-automata to DFAs or NFAs. Interestingly, we can show the tight bounds 2^n+1 and 2^n. So, the power gained by the resources given to restarting automata seems to be different compared with nondeterminism. Next, Section 4 deals with deterministic RR(1)-automata. Here we prove their computational capacity to be equivalent to finite automata. The proof concludes the distinction of restarting automata accepting regular languages. For the conversion to an NFA we show the tight trade-off of $(2n-r+2)\cdot 2^{r-1} \in O(n\cdot 2^n)$, where r is the number of different states reachable after a rewrite step.

2. Restarting automata

For an alphabet A, let A^+ be the set of nonempty words w over A. If the empty word λ is included, then we use the notation A^*. For the length of w we write $|w|$. The set of words of length $n \geqslant 0$ is denoted by A^n. If n is an upper bound for the length, we write $A^{\leqslant n}$, which is defined to be $\bigcup_{i=0}^{n} A^i$.

Concerning the notions and definitions of restarting automata, we follow the presentation in [17]. Let T be some tape alphabet, and $\rhd$ and $\lhd$ the left and right endmarker of the workspace, then the possible contents W_k of a read-write window of size $k \geqslant 2$ are $W_k = (\rhd T^{k-1}) \cup T^k \cup (T^{\leqslant k-1}\lhd) \cup (\rhd T^{\leqslant k-2}\lhd)$. For $k=1$, we have $W_1 = \{\rhd\} \cup T \cup \{\lhd\}$.

A *nondeterministic restarting automaton* (RRWW-automaton) is a system $M = \langle S,A,T,\rhd,\lhd,s_0,k,\delta\rangle$, where S is the finite set of internal states, A is the finite set of input symbols, T is the finite set of tape symbols containing A, $\rhd \notin T$ is the left and $\lhd \notin T$ is the right endmarker of the workspace, $s_0 \in S$ is the initial state, $k \geqslant 1$ is the size of the read-write window, and δ is the partial transition function mapping $S \times W_k$ to the finite subsets of $\left(S \times (\bigcup_{i=1}^{k-1} W_i \cup \{\lambda,\mathsf{MVR}\})\right) \cup \{\mathsf{Restart},\mathsf{Accept}\}$, where δ satisfies the following condition: If $(q',v_1 \ldots v_i) \in \delta(q,u_1 \ldots u_j)$, $u_1,\ldots,u_j,v_1,\ldots,v_i \in T \cup \{\rhd,\lhd\}$, then (i) $i < j$, (ii) $u_1 = \rhd$ if and only if $v_1 = \rhd$, and (iii) $u_j = \lhd$ if and only if $v_i = \lhd$.

The transition function allows four different types of steps. Let M be in a state s with $u \in W_k$ in its read-write window. A *move-right step* of the form $(s',\mathsf{MVR}) \in \delta(s,u)$ is applicable if $u \neq \lhd$. It causes M to shift the read-write window one position to the right and to enter state $s' \in S$. A *rewrite step* of the form $(s',v) \in \delta(s,u)$ is applicable if $u \neq \lhd$. It causes M to replace the content of the read-write window by v, to enter state s', and to place the read-write window immediately to the right of v. If u (and v) ends with $\lhd$, the read-write window is placed on $\lhd$. A *restart step* of the form $\mathsf{Restart} \in \delta(s,u)$ causes M to place the read-write window back on the left end of the tape such that $\rhd$ appears as the leftmost symbol in the window, and to enter the initial state s_0. An *accept step* of the form $\mathsf{Accept} \in \delta(s,u)$ causes M to halt, and to accept.

Each computation of M proceeds in *cycles*. Starting from an initial configuration $s_0\rhd w\lhd$, the window moves right until a restart step takes M back into a configuration of the form $s_0\rhd w'\lhd$. It is required that in each cycle *exactly one* rewrite step is performed. The part of the computation that follows the last restart step is called the *tail* of the computation. It contains at most one rewrite step. We denote the transition from some configuration to a successor configuration by $\vdash$, and write $\vdash^*$ for the reflexive and transitive closure of that relation. An input $w \in A^*$ is accepted by M, if there is a computation which starts with the initial configuration $s_0\rhd w\lhd$ and ends with an accept step. By $L(M)$ we denote the *language accepted* by M. Similar notions are used for other types of acceptors, too.

We omit an R of the type of restarting automata and obtain R(W)(W)-automata, if a restart step is conjoint with a rewrite step, i.e., whenever a rewrite step is performed the read-write window is placed back on the left end of the tape, and the initial state s_0 is entered. We omit one W obtaining R(R)W-automata, if the tape alphabet is equal to the input alphabet, and omit both W obtaining R(R)-automata, if the string v of each rewrite step $(s',v) \in \delta(s,u)$ is a scattered subword of u, i.e., it is obtained by deleting some symbols from u. We use the prefix det- in order to denote deterministic automata.

As mentioned in the introduction, we are interested in weak restarting automata, i.e., restarting automata that accept exactly the regular languages. In view of the nonrecursive trade-offs between different classes of restarting automata, it seems that we need a finer control of the descriptional capacity than offered by controlling the resources nondeterminism or the general write and read behavior. A good candidate is the size of the read-write window. The window size can be seen as the length of the lookahead. Related studies concern the trade-offs between $LR(k)$ and $LR(k+1)$ [13], and between $LL(k)$ and $LL(k+1)$ [1] grammars, respectively. In [14] restarting automata with bounded lookahead are studied from a computational capacity point of view. In particular, it has been shown that for all $k \geqslant 1$, there exists a language L accepted by some deterministic R-automaton with window size $k+1$ such that L is not accepted by any nondeterministic RRW-automaton with window size k. This result implies infinite lookahead hierarchies for the variants of restarting automata that