Theoretical Computer Science 526 (2014) 97–107

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

A note on the emptiness problem for alternating finite-memory automata

Daniel Genkin *, Michael Kaminski, Liat Peterfreund

Department of Computer Science, Technion – Israel Institute of Technology, Haifa 32000, Israel

ARTICLE INFO

Article history:
Received 4 March 2013
Received in revised form 7 January 2014
Accepted 15 January 2014
Communicated by D. Perrin

Keywords:
Infinite alphabets
Alternating finite-memory automata
Alternating finite-memory tree automata
Emptiness problem

ABSTRACT

We present alternative relatively simple and self-contained proofs of decidability of the emptiness problems for one-register alternating finite-memory automata and one-register alternating finite-memory tree automata.

1. Introduction

Finite-memory automata [5,6] are a generalization of the classical Rabin–Scott finite-state automata [12] to *infinite* alphabets. In addition to a finite set of "proper" states, finite-memory automata are equipped with a finite set of registers which in any stage of a computation (automaton's run) contain a symbol from the infinite alphabet. By restricting the power of the automaton to comparing the input symbol with the contents of the registers and copying the input symbol to a register, without the ability to perform *any* operations, the automaton is only able to "remember" a finite set of input symbols. Thus, the languages accepted by finite-memory automata possess many of the desirable properties of regular languages and therefore, are referred to as *quasi-regular* languages.

An important facet of finite-memory automata is a certain *indistinguishability* view of the infinite alphabet embedded in the modus operandi of the automaton. The language accepted by an automaton is invariant under automorphisms of the infinite alphabet. Thus, the actual symbols occurring in the input are of no real significance. Only the initial and repetition patterns matter. This follows from the nature of the restriction to copying and comparison (reminiscent of *term-unification*). If a new symbol (i.e., one not in any register) is copied and later successfully compared, any other new symbol, appearing in the same position, would cause the *same* transitions.

This paper deals with the *alternating* versions of finite-memory automata [10,11] and finite-memory tree automata [4]. These automata are tightly related to the linear and branching temporal logics with the freeze quantifier, see [2,3] and [4], respectively. Whereas the emptiness problem for two-register alternating finite-memory automata is undecidable, see [11, Theorem 5.1], it was shown in [2,3] and [4], respectively, that the emptiness problem for both one-register alternating finite-memory automata and one-register alternating finite-memory tree automata is decidable.[1] The proofs in [2–4] are based on reduction to decidability of the empriness problem for *counter automata with incrementing errors* and are very

* Corresponding author.

[1] Decidability of the former problem implies decidability of the emptiness problem for *top-view* 2-pebble automata, see [13,14].

0304-3975/$ – see front matter
http://dx.doi.org/10.1016/j.tcs.2014.01.020

involved.[2] In this paper we present alternative relatively simple and self-contained proofs of decidability of the emptiness problems for one-register alternating finite-memory automata and for one-register alternating finite-memory tree automata.

Theorem 1. *The emptiness problem for one-register alternating finite-memory automata is decidable.*

Theorem 2. *The emptiness problem for one-register alternating finite-memory tree automata is decidable.*

The proofs of these theorems are modifications of the decidability of the universality problem in [6, Appendix A] that is, actually, the emptiness problem for *universal* alternating finite-memory automata, whose behavior is deterministic. For the proofs of the above theorems we extend the construction in [6] to a nondeterministic behavior of general alternating finite-memory automata and alternating finite-memory tree automata. Namely, we show that, if the language of a one-register alternating finite-memory automaton (respectively, a one-register alternating finite-memory tree automaton) is nonempty, then it contains a "short" word (respectively, a "small" tree) whose length (respectively, size) is computable. Also, similarly to the construction in [6, Appendix A], our proofs extend to a proof of decidability of inclusions of a quasi-regular language in the language of a one-register alternating finite-memory automaton and a tree automaton language in the language of a one-register alternating finite-memory tree automaton.

The rest of this paper is organized as follows. In the next section we recall the definition of one-register alternating finite-memory automata and prove its basic properties needed for the proof of Theorem 1. The proof of the theorem is presented in Section 3. In Section 4, we recall the definition of one-register alternating finite-memory tree automata. Finally, Section 5 contains the proof of Theorem 2.

2. Alternating finite-memory automata

In this section we recall the definition of alternating finite-memory automata and prove its basic properties needed for the proof of Theorem 1. We restrict ourselves to one-register automata with which we deal in this paper. Like in the classical finite alphabet case, these automata have a finite set of proper states in which the "real computation" is done. In addition, the automaton is equipped with a register storing a symbol from the infinite alphabet. We refer to the register as the "*window*," see [5,6].

Throughout this paper we use the following conventions.

- Σ is a fixed infinite alphabet.
- Symbols in Σ are denoted by σ (sometimes indexed or primed), τ, or δ.
- Bold low-case Greek letters $\boldsymbol{\sigma}$, $\boldsymbol{\sigma}'$, and $\boldsymbol{\sigma}''$ denote words over Σ.
- Symbols which occur in a word denoted by a boldface letter are always denoted by the same *non-boldface* letter with an appropriate subscript. For example, symbols which occur in $\boldsymbol{\sigma}'$ are denoted by σ'_i.

Definition 3. (Cf. [6, Definition 1].) A *one-window alternating finite-memory automaton* (over Σ) is a system $\boldsymbol{A} = \langle S, s_0, F, \Delta, \delta, \mu_\Delta, \mu_=, \mu_{\neq}\rangle$ whose components are as follows.

- S is a finite set of *states*.
- $s_0 \in S$ is the *initial state*.
- $F \subseteq S$ is the set of *accepting* states.
- $\Delta \subset \Sigma$ is a finite set of *distinguished* symbols.
- $\delta \in \Sigma \setminus \Delta$ is the *initial window assignment*.
- $\mu_\Delta : S \times \Delta \to 2^{2^S}$, $\mu_= : S \to 2^{2^S}$, and $\mu_{\neq} : S \to 2^{(2^S)^2}$ are the *transition* functions.
  The intuitive meaning of these functions is as follows. Let $\boldsymbol{A}$ read a symbol σ being in state s with a symbol τ stored in the window.
  - If $\sigma \in \Delta$, then, for some $Q \in \mu_\Delta(s, \sigma)$, the computation splits to the computations from all states in Q with τ in the window.
  - If $\sigma = \tau$, then, for some $Q \in \mu_=(s)$, the computation splits to the computations from all states in Q with the same τ in the window.
  - If $\sigma \notin \Delta \cup \{\tau\}$, then, for some $(Q', Q'') \in \mu_{\neq}(s)$, the computation splits to the computations from all states in Q' with τ in the window and the computations from all states in Q'' with the current input symbol σ in the window.

In accordance with the above intuitive meaning of the transition functions, an actual state of $\boldsymbol{A}$ is an element of $S \times (\Sigma \setminus \Delta)$, where the state component of the pair is the current state and the symbol component is the symbol stored in

[2] The converse reduction shows that the emptiness problem for one-register alternating finite-memory automata is not primitive recursive, see [3, Theorem 5.2].