Theoretical Computer Science 610 (2016) 59–77

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Boundary sets of regular and context-free languages ☆

Markus Holzer, Sebastian Jakobi

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

ARTICLE INFO

Article history:
Received 4 November 2014
Received in revised form 7 July 2015
Accepted 13 July 2015
Available online 28 July 2015

Keywords:
Regular languages
Context-free languages
Boundary sets
Computational complexity
Decidability
Valid computations

ABSTRACT

We investigate the descriptional and computational complexity of boundary sets of regular and context-free languages. For a letter a, the right (left, respectively) a-boundary set of a language L consists of the words in L whose a-predecessor or a-successor w.r.t. the prefix (suffix, respectively) relation is not in L. For regular languages described by deterministic finite automata (DFAs) we give tight bounds on the number of states for accepting boundary sets. Moreover, the question whether the boundary sets of a regular language is finite is shown to be NL-complete for DFAs, while it turns out to be PSPACE-complete for nondeterministic devices. Boundary sets for context-free languages are not necessarily context free anymore. Here we find a subtle difference of right and left a-boundary sets. While right a-boundary sets of deterministic context-free languages stay deterministic context free, we give an example of a deterministic context-free language whose a-boundary set is already non-context free. In fact, the finiteness problem for a-boundary sets of context-free languages becomes undecidable.

1. Introduction

In topology, see, e.g., [1], the boundary or frontier of a subset S of a (topological) space is the set of points which can be approached both from S and from the outside of S. Here the term approaching refers to a distance relation induced by the neighborhood property. For instance, the boundary of the half-open set $[0, 1)$ is equal to the two-point set $\{0, 1\}$. This example shows that the elements of the boundary of a set S are *not* necessarily elements of the original set S. Boundaries play an important role by the integration of real or complex valued functions in function theory. We slightly adapt the definition of a boundary or frontier set to the case of formal languages. To this end, we use the prefix- or suffix-relation on the set of words as a suitable neighborhood concept. Due to the left-to-right nature of words it is natural to define two variants of boundary sets, namely left and right versions. Roughly speaking, the right (left, respectively) a-boundary set of a language L are those words that belong to L, where the a-predecessor or the a-successor of these words w.r.t. the prefix (suffix, respectively) relation is *not* in L. By the choice of our definition the elements of a boundary of a formal language L are *always* elements of the original set L.

Why is it interesting to study boundaries of formal languages? There are two major aspects. First of all, boundaries of formal languages play an important role in the solvability of certain language equations based on almost-equivalence. Two sets are almost-equivalent, if they are equivalent up to a finite number of exceptions. Recently, this form of “equivalence”

☆ This paper is a completely revised and expanded version of a paper presented at the 16th Workshop on Descriptional Complexity of Formal Systems (DCFS) held in Turku, Finland, August 5–8, 2014.

E-mail addresses: holzer@informatik.uni-giessen.de (M. Holzer), sebastian.jakobi@informatik.uni-giessen.de (S. Jakobi).

http://dx.doi.org/10.1016/j.tcs.2015.07.032
0304-3975/

has attracted a lot of attention in a series of papers on automata with errors, see, e.g., [2–4]. Our contribution to language equations with almost-equivalence is just a first step towards a more general theory on this subject. Second, the computation of the boundary set of a language can be seen as an application of a (very complex) combined operation on just one input language. Combined operations, more precisely the descriptional complexity of combined operations, for regular languages were popularized in [5]. In fact, we will show that the left a-boundary set of a language L is equal to

$$\Big(L \setminus (a^{-1} \cdot L)\Big) \cup a \cdot \Big((a^{-1} \cdot L) \setminus L\Big);$$

a similar result is also valid for the right a-boundary set of L—here the only non-standard operation is $a^{-1} \cdot L := \{ u \in \Sigma^* \mid au \in L \}$, which refers to the *left quotient* or *left derivative* of $L \subseteq \Sigma^*$ by a letter $a \in \Sigma$.

We define the various boundary sets of languages in Section 2. There we also give an introductory example and explain the connection of boundaries of languages to language equations with almost-equivalence. Here the finiteness of a certain boundary set is a necessary precondition for the solvability of the language equation. Therefore, we later also investigate the decidability status of the finiteness problem for the boundary of languages. But before that we turn our attention to the descriptional complexity of language boundaries. For regular languages we find the following situation in Section 3: by standard automata constructions on the language operations it is easy to see that the boundary set of a regular language is regular, too. Moreover, one may deduce an upper bound of $O(n^4)$ for the a-boundary sets in general. A closer look reveals that the right a-boundary set obeys a tight linear bound on the number of states for deterministic finite automata (DFAs), while the state complexity of the left a-boundary set of a language accepted by an n-state DFA is at most $n^2 \cdot (n-1)/2 + 2$, and this bound is also tight. The study on boundary sets of regular languages continues in Section 4 by the investigation on the decidability of the finiteness problem for boundaries of regular languages. The question whether the boundary of a regular language is finite is shown to be NL-complete for DFAs, while it turns out to be PSPACE-complete for nondeterministic devices. Finally, our focus is changed to boundary sets of context-free languages. First we show in Section 5 that boundary sets for context-free languages are not necessarily context free anymore. Moreover, we also find a subtle difference of right and left a-boundary sets. While right a-boundary sets of *deterministic* context-free languages stay deterministic context free, we give an example of a deterministic context-free language the left a-boundary set of which is already not context free. In fact, the finiteness problem for a-boundary sets of context-free languages becomes undecidable. This closes our study on boundaries of regular and context-free languages. Nevertheless, some problems remain open. For instance, the decidability of the finiteness problem for left a-boundary sets of deterministic context-free languages is yet unknown.

2. Preliminaries

We recall some definitions on finite automata as contained in [6]. A *nondeterministic finite automaton* (NFA) is a quintuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q is the finite set of *states*, Σ is the finite set of *input symbols*, $q_0 \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *accepting states*, and $\delta\colon Q \times \Sigma \to 2^Q$ is the *transition function*. The *language accepted* by the finite automaton A is defined as $L(A) = \{ w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \emptyset \}$, where the transition function is recursively extended to $\delta\colon Q \times \Sigma^* \to 2^Q$. A finite automaton is *deterministic* (DFA) if and only if $|\delta(q,a)| = 1$, for all states $q \in Q$ and letters $a \in \Sigma$. In this case we simply write $\delta(q,a) = p$ instead of $\delta(q,a) = \{p\}$, assuming that the transition function is a total mapping $\delta\colon Q \times \Sigma \to Q$.

In this paper we study some aspects on the descriptional and computational complexity of language boundaries. To this end we define for a language $L \subseteq \Sigma^*$ and a letter $a \in \Sigma$, the *left a-boundary of L* as

$${}_a\partial(L) = {}_a\partial^{\uparrow}(L) \cup {}_a\partial^{\downarrow}(L),$$

where

$${}_a\partial^{\uparrow}(L) = \{ w \in L \mid aw \notin L \} \quad \text{and} \quad {}_a\partial^{\downarrow}(L) = \{ aw \in L \mid w \notin L \}.$$

Here the former set is referred to as the *upper left a-boundary of L*, while the latter set is called the *lower left a-boundary of L*. Similarly one defines the *right a-boundary of L* as the language $\partial_a(L) = \partial_a^{\uparrow}(L) \cup \partial_a^{\downarrow}(L)$, where we refer to $\partial_a^{\uparrow}(L) = \{ w \in L \mid wa \notin L \}$ as the *upper right a-boundary of the set L* and we define $\partial_a^{\downarrow}(L) = \{ wa \in L \mid w \notin L \}$ to be the *lower right a-boundary of L*. In order to clarify our notation we give an example.

Example 1. Let $\Sigma = \{a, b\}$. Consider the language $L = \{a, a^2, a^4, b, ab\}$. Then the left a-boundary of L is ${}_a\partial(L) = \{a, a^2, a^4, ab\}$ because ${}_a\partial^{\uparrow}(L) = \{a^2, a^4, ab\}$, and ${}_a\partial^{\downarrow}(L) = \{a, a^4\}$. The right a-boundary of L is $\partial_a(L) = \{a, a^2, a^4, b, ab\}$. The computation of the left (right, respectively) a-boundary set can be visualized by a left (right, respectively) language tree—see Fig. 1. Taking an edge in a left (right, respectively) language tree corresponds to a left (right, respectively) concatenation of a letter. From the left language tree one can read out the elements of the left a-boundary set ${}_a\partial(L)$ by simply collecting all words that belong to the language L but where the a-successor or -predecessor does not belong to the set under consideration. Here the a-successor of a node is its left child in the tree, and a node p is the a-predecessor of a node q, if q is the a-successor of p. A similar construction can be applied for the right a-boundary using the right language tree. □