Theoretical Computer Science 578 (2015) 88–99

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Lambda-confluence for context rewriting systems ☆

Friedrich Otto [a,*], František Mráz [b,1]

[a] Fachbereich Elektrotechnik/Informatik, Universität Kassel, 34109 Kassel, Germany
[b] Charles University, Faculty of Mathematics and Physics, Department of Computer Science, Malostranské nám. 25, 118 00 Praha 1, Czech Republic

ARTICLE INFO

Article history:
Received 28 October 2013
Received in revised form 23 May 2014
Accepted 13 January 2015
Available online 19 January 2015

Keywords:
Context rewriting system
Clearing restarting automaton
Limited context restarting automaton
Factor-erasing string-rewriting system
λ-Confluence

ABSTRACT

Clearing restarting automata and limited context restarting automata are particular types of context rewriting systems. A word w is accepted by such a system if there is a sequence of rewritings that reduces the word w to the empty word λ, where each rewrite rule is extended by certain context conditions. If each rewrite step is strictly length-reducing, as for example in the case of clearing restarting automata, then the word problem for such a system can be solved nondeterministically in quadratic time. If, in addition, the contextual rewritings happen to be λ-confluent, that is, confluent on the congruence class of the empty word, then the word problem can be solved deterministically in linear time. Here we show that λ-confluence is decidable in polynomial time for limited context restarting automata of type $\mathcal{R}_2$, but that this property is not even recursively enumerable for clearing restarting automata. The latter follows from the fact that λ-confluence is not recursively enumerable for finite factor-erasing string-rewriting systems.

1. Introduction

Restarting automata were introduced in [12] to model the technique of *analysis by reduction*, which is used in linguistics to analyze sentences of natural languages with free word order. Interestingly, a restarting automaton is not only useful for accepting a language, but it also enables error localization in rejected words (see, e.g., [11]). Despite these nice properties, restarting automata are rarely used in practice. One reason for this is certainly the fact that it is quite a complex task to design a restarting automaton for a given language. Accordingly, methods have been studied for learning a restarting automaton from positive (and negative) examples of sentences and/or reductions (see, e.g., [1,2,6–8,18]).

Specifically, Černo and Mráz introduced a restricted type of restarting automaton, the so-called clearing restarting automaton, in [8], which is a special type of context rewriting system. In general, a *context rewriting system* C is given through a triple $C = (\Sigma, \Gamma, I)$, where Σ is a finite input alphabet, Γ is a finite working alphabet that contains Σ and possibly some additional so-called auxiliary letters, but that does not contain the special symbols ¢ and \$, called sentinels, and I is a finite set of instructions of the form $(x \mid z \to t \mid y)$. Here $x \in \Gamma^* \cup \{¢\} \cdot \Gamma^*$ is the *left context*, $y \in \Gamma^* \cup \Gamma^* \cdot \{\$\}$ is the *right context*, and $(z \to t)$ is the *rewrite rule* of this instruction, where $z, t \in \Gamma^*$. The idea of the instruction $i = (x \mid z \to t \mid y)$ is that based on the local context x and y, an occurrence of the factor z can be replaced by t, that is, a factor xzy is rewritten

☆ Some of the results of this paper have been announced at the 18-th International Conference on Implementation and Application of Automata, CIAA 2013, in Halifax, Nova Scotia, Canada, July 2013. An extended abstract appeared in the proceedings of that conference [17].

* Corresponding author.
E-mail addresses: otto@theory.informatik.uni-kassel.de (F. Otto), mraz@ksvi.ms.mff.cuni.cz (F. Mráz).

[1] F. Mráz was partially supported by the Grant Agency of the Czech Republic under the project P103/10/0783.

http://dx.doi.org/10.1016/j.tcs.2015.01.013
0304-3975/

into xty. Now a word $w \in \Sigma^*$ is accepted by C, if there is a sequence of applications of instructions from I that reduces the word ¢w\$ to the word ¢\$, that is, the word w is reduced to the empty word λ in the context (¢, \$). A *clearing restarting automaton* is a context rewriting system $M = (\Sigma, \Gamma, I)$ for which Γ equals Σ, that is, no auxiliary symbols are available, and each instruction $(x \mid z \to t \mid y) \in I$ satisfies the restrictions that $z \in \Sigma^+$ and $t = \lambda$. For clearing restarting automata a simple learning algorithm exists [6,8], but on the other hand, they are quite limited in their expressive power. In fact, while these automata accept all regular languages and even some languages that are not context-free, they do not even accept all context-free languages (see [8]).

Later the clearing restarting automaton was extended to the limited context restarting automaton by Basovník and Mráz in [1,2]. A *limited context restarting automaton* (lc-R-automaton, for short) M is defined through a finite set of instructions of the form $(x \mid z \to t \mid y)$, where $|z| > |t| \geq 0$, that is, the lc-R-automata are essentially just the length-reducing context rewriting systems. Several different types of lc-R-automata have been defined in [2] and in [25] based on the form of the admissible contexts x and y and the form of the word t. In an lc-R-automaton of type $\mathcal{R}_1$, we have $|t| \leq 1$ for each instruction, and for an lc-R-automaton of type $\mathcal{R}_2$, we require in addition that $x \in \{¢, \lambda\}$ and $y \in \{\$, \lambda\}$ for all instructions, that is, the left (right) context of each instruction is either the left (right) sentinel, or it is empty. Obviously, the lc-R-automaton of type $\mathcal{R}_1$ is a proper extension of the clearing restarting automaton, while those of type $\mathcal{R}_2$ are incomparable to clearing restarting automata. As observed in [2,26], the lc-R-automata of type $\mathcal{R}_2$ accept exactly the context-free languages, while those of type $\mathcal{R}_1$ characterize the class GACSL of *growing acyclic context-sensitive languages* of Buntrock [5] (see also [19]).

To test whether a word w belongs to the language $L(M)$ accepted by a given lc-R-automaton M, one has to check whether ¢w\$ can be reduced to the empty word ¢\$ by a sequence of applications of instructions of M. As each instruction is length-reducing, such a sequence is bounded in length by $|w|$, but as there could be several instructions that are applicable to the same word, or there could be several places at which a given instruction can be applied, all such sequences must be checked. Accordingly, the membership problem for $L(M)$ is decidable nondeterministically in time $O(n^2)$. The situation would be much better if it was known that each and every sequence of applications of instructions of M reduces ¢w\$ to ¢\$, if w does indeed belong to the language $L(M)$. In this case we could concentrate on leftmost sequences of reductions, and accordingly, membership in $L(M)$ would be decidable deterministically in time $O(n)$.

With a context rewriting system $C = (\Sigma, \Gamma, I)$, we can associate a finite string-rewriting system $S_0(C) = \{xzy \to xty \mid (x \mid z \to t \mid y) \in I\} \cup \{¢\$ \to \lambda\}$. Obviously, for all input words w, w is accepted by C if and only if ¢w\$ $\Rightarrow^*_{S_0(C)} \lambda$ holds, where $\Rightarrow^*_{S_0(C)}$ denotes the reduction relation induced by $S_0(C)$ (see below). Now the context rewriting system C is called *confluent*, if the string-rewriting system $S_0(C)$ is confluent. This means that there is at most a single irreducible word $\hat{u}$ modulo $S_0(C)$ for each word $u \in (\Gamma \cup \{¢, \$\})^*$ (see, e.g., [4]). As it turned out, however, confluent lc-R-automata of type $\mathcal{R}_1$ ($\mathcal{R}_2$) are much less expressive than the non-confluent lc-R-automata of the same type [25,26]. In fact, confluent lc-R-automata of type $\mathcal{R}_1$ can only accept Church–Rosser languages (see [16]), while confluent lc-R-automata of type $\mathcal{R}_2$ can only accept confluent monadic McNaughton languages (see [3]), and in both cases it is open whether the indicated inclusion is proper or not.

However, for solving the membership problem for the language $L(M)$ of an lc-R-automaton M in linear time, confluence of $S_0(M)$ is actually not needed. In fact, it would suffice that each word of the form ¢w\$, which is congruent to the word ¢\$, can be reduced to ¢\$ by rewriting with respect to the system $S(M) = S_0(M) \setminus \{¢\$ \to \lambda\}$. In this case we say that the lc-R-automaton M is λ-*confluent*. In [7] Černo presents an inductive inference algorithm for context rewriting systems that works in polynomial time for λ-confluent lc-R-automata.

An lc-R-automaton M is in particular λ-confluent, if the string-rewriting system $S(M)$ is confluent on the congruence class of the word ¢\$, which means that each word $u \in (\Gamma \cup \{¢, \$\})^*$ that is congruent to ¢\$ mod $S(M)$ can actually be reduced to ¢\$ by applying the rules of $S(M)$. However, this property is still too restrictive, as we will see in Example 3.1 below that an lc-R-automaton M can be λ-confluent even if the corresponding string-rewriting system $S(M)$ is not confluent on the congruence class of the word ¢\$.

Here we study the problem of deciding whether a given lc-R-automaton is λ-confluent. For finite string-rewriting systems, this problem has received quite some attention before. In [23] it was shown that confluence on a given congruence class (and in particular, λ-confluence) is undecidable for finite length-reducing string-rewriting systems, and that this problem is decidable in double exponential time for finite monadic string-rewriting systems (see Section 2 for the definitions). Further, it was shown in [24] that this problem is decidable in polynomial time for finite special string-rewriting systems, and then Sénizergues improved the complexity result for finite monadic string-rewriting systems by showing that this problem is actually decidable in polynomial time as well [27]. In fact, his result is slightly stronger as it considers the property of confluence not just on a single congruence class, but on a given regular set of irreducible words (that is, representatives of congruence classes), and he considers finite, basic, noetherian string-rewriting systems, which form a proper superclass of the finite monadic string-rewriting systems.

If M is a clearing restarting automaton, then each rule of the string-rewriting system $S(M)$ simply erases a non-empty factor of its left-hand side. Thus, in this case $S(M)$ is a *factor-erasing* string-rewriting system. As our technical main result we will show that it is undecidable in general whether a given finite factor-erasing string-rewriting system is λ-confluent. In fact, we will see that this problem is not even recursively enumerable (r.e.). By associating a clearing restarting automaton M_S with each finite factor-erasing string-rewriting system S such that M_S is λ-confluent if and only if S is λ-confluent, it is then shown that it is undecidable (in fact, not r.e.) in general whether a given clearing restarting automaton is λ-confluent. As clearing restarting automata are lc-R-automata of type $\mathcal{R}_1$, this result extends to lc-R-automata of type $\mathcal{R}_1$. On the other hand, we will see that λ-confluence is decidable in polynomial time for lc-R-automata of type $\mathcal{R}_2$ by reducing this problem