Theoretical Computer Science 207 (1998) 43–72

Theoretical Computer Science

ELSEVIER

Some undecidability results concerning the property of preserving regularity [1]

Friedrich Otto *

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

Abstract

A finite string-rewriting system R preserves regularity if and only if it preserves Σ-regularity, where Σ is the alphabet containing exactly those letters that have occurrences in the rules of R. This proves a conjecture of Gyenizse and Vágvölgyi (1997). In addition, some undecidability results are presented that generalize results of Gilleron and Tison (1995) from term-rewriting systems to string-rewriting systems. It follows that the property of being regularity preserving is undecidable for term-rewriting systems, thus answering another question of Gyenizse and Vágvölgyi (1997). Finally, it is shown that it is undecidable in general whether a finite, length-reducing, and confluent string-rewriting system yields a regular set of normal forms for each regular language. © 1998—Elsevier Science B.V. All rights reserved

Keywords: Rewriting systems; String-rewriting; Sets of descendants; Sets of normal forms; Regularity preserving

1. Introduction

In the specification of abstract data types the use of rewriting techniques is by now well established. In this context the initial algebra defined by a given set of equations or rewrite rules is of particular interest. It is the set of ground terms in the signature considered modulo the congruence that is generated by the given equations or rules (see, e.g., [27]).

A set of ground terms is a tree language. A class of tree languages that has been studied in great detail is the class of regular tree languages. This class is defined by finite tree automata, and it enjoys many closure properties as well as decidability results in common with the class of regular string languages [9, 10].

[1] Some of the results presented here have been announced at the 8th International Conference on Rewriting Techniques and Applications (RTA) at Sitges, Spain, June 1997.

* E-mail: otto@theory.informatik.uni-kassel.de.

PII S0304-3975(98)00055-3

If R is a left-linear term-rewriting system on a signature F, then the set of normal forms $\mathrm{IRR}(R)$ is a regular tree language [8]. The system R is called *F-regularity preserving*, if, for each regular tree language $S \subseteq T(F)$, the set of descendants $\Delta_R^*(S)$ is again a regular tree language. Thus, if R is a left-linear and convergent term-rewriting system that is F-regularity preserving, then, for each regular tree language $S \subseteq T(F)$, the set of normal forms $\mathrm{NF}_R(S) = \Delta_R^*(S) \cap \mathrm{IRR}(R)$ of ground terms that are in normal form and that are congruent to some ground term in S is a regular tree language. Hence, for systems of this form, various decision problems can be solved efficiently by using tree automata (see, e.g., [9, 10, 12]).

It is known that F-regularity is preserved by term-rewriting systems that contain only ground rules [5], by term-rewriting systems that are right-linear and monadic [25], that are linear and semi-monadic [6], or that are linear and generalized semi-monadic [14]. Recall that the monadic term-rewriting systems were introduced by Book and Gallier as a direct generalization of the monadic string-rewriting systems [8].

However, the property of preserving F-regularity is undecidable in general. This already follows from the construction used by Dauchet in [7] to prove that termination is undecidable for left-linear one-rule term-rewriting systems, but a less complicated direct proof is given in [12]. In addition, also the property that $\mathrm{NF}_R(S)$ is regular for each regular tree language $S \subseteq T(F)$ is undecidable in general [11]. Actually, the latter undecidability result remains valid even for the class of finite term-rewriting systems that are convergent [11, 12].

It has been observed by Gyenizse and Vágvölgyi that the property of preserving F-regularity does not only depend on the term-rewriting system R considered, but also on the actual signature F being used [14]. In fact, they present a signature F that contains a single constant plus some unary function symbols only, and a term-rewriting system R over F that consists of a single linear rule plus some ground rules such that R preserves F-regularity, but R does not preserve F_1-regularity, where the signature F_1 is obtained from F by introducing an additional unary function symbol. Accordingly, they call a system R on a signature F *regularity preserving*, if it is F_1-regularity preserving for each signature F_1 containing F. They show that the property of being regularity preserving is a modular property of linear term-rewriting systems, and they ask whether this property is undecidable in general.

Since each string-rewriting system can be interpreted as a linear term-rewriting system, it follows that for string-rewriting systems the property of being regularity preserving is modular. Gyenizse and Vágvölgyi conjecture that for string-rewriting systems the properties of preserving F-regularity and of preserving regularity are equivalent. Here we prove this conjecture.

If R is a string-rewriting system on some alphabet Σ, then with R we can associate a linear term-rewriting system S_R over the signature F_Σ, which contains unary function symbols and a single constant only. We will see that R preserves regularity if and only if the term-rewriting system S_R does. Since the property of preserving regularity is undecidable for finite string-rewriting systems, this shows that the property of being regularity preserving is undecidable for linear term-rewriting systems, thus answering the question of Gyenizse and Vágvölgyi mentioned above.

For string-rewriting systems the property of being a generalized semi-monadic system is just the property of being a left-basic string-rewriting system (see, e.g., [26]), and it is known that a finite, length-reducing, confluent, and left-basic string-rewriting system R yields a regular set $\mathrm{NF}_R(S)$ for each regular language S [24]. Thus, for string-rewriting systems the above-mentioned result that a linear, generalized semi-monadic system preserves F-regularity can be seen as a generalization of Sakarovitch's result. On the other hand, it is known that each monadic string-rewriting system preserves regularity [3, 18, 25]. Indeed, this fact has been exploited by Book to show that for a finite, monadic, and confluent string-rewriting system all those properties of the Thue congruence generated are decidable that can be expressed by linear sentences [2]. However, there exist finite, length-reducing, and confluent string-rewriting systems that do not preserve regularity [21]. In fact, there exist systems R of this form and regular languages S such that the sets of descendants $\Delta_R^*(S)$ are arbitrarily complex. Actually, for each recursively enumerable language $L \subset \Sigma^*$, there exist a finite, length-reducing, and confluent string-rewriting system R on some alphabet $\Gamma \supset \Sigma$ and two regular languages $S_1, S_2 \subset \Gamma^*$ such that $\pi_\Sigma(\Delta_R^*(S_1) \cap S_2) = L$ holds, where π_Σ denotes the projection from Γ^* onto Σ^* [22].

Here we give a simple proof for the fact that the property of preserving regularity is undecidable for finite string-rewriting systems in general. Further, we construct a finite, length-reducing, and confluent string-rewriting system R such that it is undecidable whether, for a regular language $S \subseteq \Sigma^*$, the set $\Delta_R^*(S)$ is a regular language. The same result is also established for the set of normal forms $\mathrm{NF}_R(S)$. In addition, a finite, length-reducing, and confluent string-rewriting system R and an infinite family of regular languages $(S_i)_{i \in \mathbb{N}}$ are constructed such that, for each $i \in \mathbb{N}$, the set of normal forms $\mathrm{NF}_R(S_i)$ is a singleton, but it is undecidable in general whether $\Delta_R^*(S_i)$ is a regular language.

Finally, using the undecidability of the *strong boundedness* of single-tape Turing machines we prove that it is undecidable in general whether a finite, length-reducing, and confluent string-rewriting system R yields a regular set of normal forms $\mathrm{NF}_R(S)$ for each regular language S. Technically this proof is by far the most involved one presented here.

This paper is organized as follows. In Section 2 we restate the basis definitions used in short in order to establish notation. In Section 3 we prove the conjecture of Gyenizse and Vágvölgyi stating that for string-rewriting systems the property of preserving regularity is independent of the alphabet actually considered, and we derive the undecidability of the property of preserving regularity for linear term-rewriting systems. In Section 4 we present the first of the undecidability results for string-rewriting systems mentioned above. Then in Section 5 we consider the strong boundedness problem for single-tape Turing machines, and finally in Section 6 we reduce this problem to the problem of deciding whether a finite, length-reducing, and confluent string-rewriting system yields a regular set of normal forms for each regular language. The paper closes with a short discussion of some open problems.

2. String-rewriting systems and monoid-presentations

After establishing notation we describe the problems considered in this paper in detail. For more information and a detailed discussion of the notions introduced the reader is refered to the literature, e.g., [4].

Let Σ be a finite alphabet. Then Σ^* denotes the set of all strings over Σ including the empty string λ. For $u, v \in \Sigma^*$, the concatenation of u and v is simply written as uv, and exponents are used to abbreviate strings, that is, $u^0 := \lambda$, $u^1 := u$, and $u^{n+1} := u^n u$ for all $n \geqslant 1$. For $u \in \Sigma^*$, the *length* of u is denoted by $|u|$.

A *string-rewriting system* R on Σ is a subset of $\Sigma^* \times \Sigma^*$, the elements of which are called *(rewrite) rules*. Often these rules will be written in the form $\ell \to r$ to improve readability. For a string-rewriting system R, $\mathrm{dom}(R) := \{\ell \mid \exists r \in \Sigma^* : (\ell \to r) \in R\}$ is the *domain* of R. The system R is called *length-reducing* if $|\ell| > |r|$ holds for each rule $(\ell \to r)$ of R, and it is called *monadic* if it is length-reducing, and $r \in \Sigma \cup \{\lambda\}$ for each rule $(\ell \to r)$ of R.

The *single-step reduction relation* induced by R is the following binary relation on Σ^* :

$$u \to_R v \text{ iff } \exists x, y \in \Sigma^* \exists (\ell \to r) \in R : u = x\ell y \text{ and } v = xry.$$

Its reflexive and transitive closure $\to_R^*$ is the *reduction relation* induced by R. The reflexive, symmetric, and transitive closure $\leftrightarrow_R^*$ of $\to_R$ is a congruence on Σ^*, the *Thue congruence* generated by R. For $u \in \Sigma^*, \Delta_R^*(u) := \{v \in \Sigma^* \mid u \to_R^* v\}$ is the *set of descendants* of $u \pmod R$, and $[u]_R := \{v \in \Sigma^* \mid u \leftrightarrow_R^* v\}$ is the *congruence class* of $u \pmod R$. For $S \subseteq \Sigma^*, \Delta_R^*(S) = \bigcup_{u \in S} \Delta_R^*(u)$ and $[S]_R := \bigcup_{u \in S} [u]_R$.

Let Σ be a finite alphabet, and let R be a string-rewriting system on Σ. We say that R *preserves Σ-regularity* if $\Delta_R^*(S)$ is a regular language for each regular language $S \subseteq \Sigma^*$. We say that R *preserves regularity* if R preserves Γ-regularity for each finite alphabet Γ containing all the letters that have occurrences in the rules of R. In the next section we will investigate the relationship between the property of preserving regularity and the property of preserving Σ-regularity, where Σ is the smallest alphabet that contains all the letters with occurrences in R. After that we will address the following decision problems:

Problem 1 (*Preserving regularity*)
Instance: A finite string-rewriting system R.
Question: Does R preserve regularity?

Problem 2 (*Regular descendants for a given language*)
Instance: A finite string-rewriting system R on Σ, and a regular language $S \subseteq \Sigma^*$.
Question: Is $\Delta_R^*(S)$ a regular language?

A string $u \in \Sigma^*$ is called *reducible* (mod R), if there exists a string $v \in \Sigma^*$ such that $u \to_R v$; otherwise, u is called *irreducible*. By IRR(R) we denote the set of all irreducible strings, and by RED(R) we denote the set of strings that are reducible (mod R). Obviously, $\mathrm{RED}(R) = \Sigma^* \cdot \mathrm{dom}(R) \cdot \Sigma^*$ and $\mathrm{IRR}(R) = \Sigma^* \setminus \mathrm{RED}(R)$. Thus, if R is finite, then RED(R) and IRR(R) are both regular sets. Actually, from R deterministic finite-state acceptors (dfsa) can easily be constructed for IRR(R) and for RED(R) (cf., e.g., Lemma 2.1.3 of [4]).

If $u \in \Sigma^*$ and $v \in \mathrm{IRR}(R)$ such that $u \to_R^* v$, then v is called a *normal form* of u. Accordingly, $\Delta_R^*(u) \cap \mathrm{IRR}(R)$ is the set of all normal forms of u, and for $S \subseteq \Sigma^*, \Delta_R^*(S) \cap \mathrm{IRR}(R)$ is the set of all normal forms of S. If a finite string-rewriting system R preserves regularity, then the set $\mathrm{NF}_R(S) := \Delta_R^*(S) \cap \mathrm{IRR}(R)$ of normal forms of S is a regular language for each regular language $S \subseteq \Sigma^*$. On the other hand, $\mathrm{NF}_R(S)$ can be a regular language, even if $\Delta_R^*(S)$ is not. Thus, also the following decision problems are of interest:

Problem 3 (*Regular sets of normal forms*)
Instance: A finite string-rewriting system R on Σ.
Question: Is $\mathrm{NF}_R(S)$ a regular language for each regular language $S \subseteq \Sigma^*$?

Problem 4 (*Regular set of normal forms for a given language*)
Instance: A finite string-rewriting system R on Σ, and a regular language $S \subseteq \Sigma^*$.
Question: Is $\mathrm{NF}_R(S)$ a regular language?

We need a couple of more definitions. Since $\leftrightarrow_R^*$ is a congruence, the set $M_R := \{[u]_R \mid u \in \Sigma^*\}$ of congruence classes mod R is a monoid under the operation $[u]_R \circ [v]_R = [uv]_R$ with identity $[\lambda]_R$. It is the factor monoid $\Sigma^*/\leftrightarrow_R^*$, and if M is a monoid that is isomorphic to M_R, then the ordered pair $(\Sigma; R)$ is called a *monoid-presentation* of M with *generators* Σ and *defining relations* R.

Some of the monoids that we will encounter are actually groups. Although groups can be described by monoid-presentations, they are usually defined through so-called group-presentations. A finite *group-presentation* $\langle \Sigma; L \rangle$ consists of a finite alphabet Σ and a finite set of defining relators $L \subseteq \underline{\Sigma}^*$, where $\overline{\Sigma}$ is an alphabet in one-to-one correspondence to Σ such that $\Sigma \cap \overline{\Sigma} = \emptyset$, and $\underline{\Sigma} := \Sigma \cup \overline{\Sigma}$. The group G_L defined by $\langle \Sigma; L \rangle$ coincides with the monoid M_{R_L} that is given through the monoid-presentation $(\underline{\Sigma}; R_L)$, where $R_L := \{a\bar{a} \to \lambda, \bar{a}a \to \lambda \mid a \in \Sigma\} \cup \{u \to \lambda \mid u \in L\}$. It is easily verified that the monoid M_{R_L} is indeed a group.

We are in particular interested in the decision problems above for those finite string-rewriting systems that are convergent. Here a string-rewriting system R on Σ is called

- *noetherian*, if there is no infinite sequence of reductions of the form $u_0 \to_R u_1 \to_R u_2 \to_R \cdots \to_R u_i \to_R u_{i+1} \to_R \cdots$;
- *confluent*, if, for all $u, v, w \in \Sigma^*, u \to_R^* v$ and $u \to_R^* w$ imply that $v \to_R^* z$ and $w \to_R^* z$ hold for some $z \in \Sigma^*$;
- *convergent*, if it is noetherian and confluent.

If R is convergent, then each congruence class $[u]_R$ contains a unique irreducible string u_0, and u_0 can be obtained from u by a finite sequence of reduction steps. Thus, the word problem is decidable for each finite convergent string-rewriting system.

Obviously, a length-reducing string-rewriting system is noetherian. The finite, length-reducing, and confluent string-rewriting systems are of particular interest, since their word problems are even solvable in linear time [4].

3. Independence of the alphabet considered

A signature F consists of a (finite) set of function symbols, each equipped with a fixed arity. If $F := \{f, g, a\}$, where f and g are unary symbols and a is a symbol of arity 0 (a constant), then for $R := \{f(g(x)) \to f(f(g(g(x)))), f(a) \to a, g(a) \to a, a \to f(a), a \to g(a)\}$ it is easily seen that $\Delta_R^*(t) = T(F)$ holds for all ground terms $t \in T(F)$. However, if $F_1 := F \cup \{h\}$, where h is another unary function symbol, then $\Delta_R^*(\{f(g(h(a)))\}) = \{f^n(g^n(h(t))) \mid t \in T(F)\}$, which is not regular. Thus, R does preserve F-regularity, but not F_1-regularity [14]. Obviously the ground rules contained in R are responsible for this, since the subsystem $R' := \{f(g(x)) \to f(f(g(g(x))))\}$ of R does not even preserve F-regularity.

Actually Gyenizse and Vágvölgyi conjecture that this phenomenon does not occur with string-rewriting systems. Here we provide a proof for this conjecture. It is a consequence of the following technical result.

Lemma 3.1. *Let R be a string-rewriting system on some finite alphabet Σ, let a be an additional letter not in Σ, and let $\Gamma := \Sigma \cup \{a\}$. If R preserves Σ-regularity, then it also preserves Γ-regularity.*

Proof. Let $S \subseteq \Gamma^*$ be a regular language. We must verify that, under the hypothesis that R preserves Σ-regularity, $\Delta_R^*(S) := \{v \in \Gamma^* \mid \exists u \in S : u \to_R^* v\}$ is a regular language.

If $S \subseteq \Sigma^*$, then $\Delta_R^*(S) \subseteq \Sigma^*$, and hence, $\Delta_R^*(S)$ is a regular language by our assumption on R. So let us assume that $S \not\subseteq \Sigma^*$. Since $S \subseteq \Gamma^*$ is a regular language, there exists an incomplete deterministic finite state acceptor $A = (Q, \Gamma, q_0, F, \delta)$ such that $L(A) = S$, and all the states of A are accessible as well as co-accessible. Let

$$(1)\ q_1 \xrightarrow{a} p_1,\quad (2)\ q_2 \xrightarrow{a} p_2, \ldots,\quad (m)\ q_m \xrightarrow{a} p_m$$

be the set of all a-transitions of A, which are numbered in an arbitrary, but fixed way. Based on A we define some auxiliary languages:

$$\begin{aligned}
S_0 &:= \{w \in \Sigma^* \mid \delta(q_0, w) \in F\} = S \cap \Sigma^*,\\
P_i &:= \{w \in \Sigma^* \mid \delta(q_0, w) = q_i\},\ \ i = 1, \ldots, m,\\
S_j &:= \{w \in \Sigma^* \mid \delta(p_j, w) \in F\},\ \ j = 1, \ldots, m,\\
K_{i,j,k} &:= \{w \in \Gamma^* \mid \delta(p_i, w) = q_j \text{ and, for all } u, v \in \Gamma^*, \text{ if } w = uav, \text{ then}\\
&\qquad \delta(p_i, u) = q_\ell \text{ for some } \ell \leqslant k\},\quad i, j = 1, \ldots, m, k = 0, 1, \ldots, m.
\end{aligned}$$

Thus, $w \in K_{i,j,k}$ if and only if $\delta(p_i, w) = q_j$, and the only a-transitions that are used when following the computation of $\delta(p_i, w)$ are those with index at most k.

Obviously, the languages S_0, P_i $(i = 1, \dots, m)$, and S_j $(j = 1, \dots, m)$ are regular.

Claim 1. *The languages $K_{i,j,k}$ are regular $(i,j = 1, \dots, m,\ k = 0, 1, \dots, m)$.*

Proof. We proceed by induction on k.

k = 0: $K_{i,j,0} = \{w \in \Gamma^* \mid \delta(p_i, w) = q_j$, and no a-transitions are used$\}$
$= \{w \in \Sigma^* \mid \delta(p_i, w) = q_j\}$.
These languages are certainly regular.

k = 1: $K_{i,j,1} = \{w \in \Gamma^* \mid \delta(p_i, w) = q_j$, and only the a-transition (1) $q_1 \xrightarrow{a} p_1$ may be used$\}$
$= K_{i,j,0} \cup K_{i,1,0} \cdot (\{a\} \cdot K_{1,1,0})^* \cdot \{a\} \cdot K_{1,j,0}$.
From the case $k = 0$ we conclude that these languages are regular.

$k \longrightarrow k$+1: $K_{i,j,k+1} = \{w \in \Gamma^* \mid \delta(p_i, w) = q_j$, and only the a-transitions (1) $q_1 \xrightarrow{a} p_1, \dots,$ (k+1) $q_{k+1} \xrightarrow{a} p_{k+1}$ may be used$\}$
$= K_{i,j,k} \cup K_{i,k+1,k} \cdot (\{a\} \cdot K_{k+1,k+1,k})^* \cdot \{a\} \cdot K_{k+1,j,k}$.
From the induction hypothesis we see that these languages are regular.

This completes the proof of Claim 1. □

In fact, it is easily seen that $K_{i,j,k}$ is accepted by the automaton $(Q, \Gamma, p_i, \{q_j\}, \delta_k)$, where δ_k is obtained from δ by removing the a-transitions with index larger than k. However, we will need the inductive representation of the sets $K_{i,j,k}$ developed above at a later stage of the proof of Lemma 3.1.

It is easily seen that

$$S = S_0 \cup \bigcup_{i=1}^{m} (P_i \cdot \{a\} \cdot S_i) \cup \bigcup_{i,j=1}^{m} (P_i \cdot \{a\} \cdot K_{i,j,m} \cdot \{a\} \cdot S_j).$$

Actually, since A is deterministic, this is a disjoint partitioning of S.

The next claim states that the operations of union and of computing descendants mod R commute.

Claim 2. $\Delta_R^*(\bigcup_{i \in I} M_i) = \bigcup_{i \in I} \Delta_R^*(M_i)$ *for all* $M_i \subseteq \Gamma^* (i \in I)$.

Proof. Since $M_i \subseteq \bigcup_{i \in I} M_i$, $\Delta_R^*(M_i) \subseteq \Delta_R^*(\bigcup_{i \in I} M_i)$, and hence, $\bigcup_{i \in I} \Delta_R^*(M_i) \subseteq \Delta_R^*(\bigcup_{i \in I} M_i)$. Conversely, if $w \in \Delta_R^*(\bigcup_{i \in I} M_i)$, then there exists some $i \in I$ such that $w \in \Delta_R^*(M_i)$. Thus, $\Delta_R^*(\bigcup_{i \in I} M_i) = \bigcup_{i \in I} \Delta_R^*(M_i)$. □

Hence, $\Delta_R^*(S) = \Delta_R^*(S_0) \cup \bigcup_{i=1}^{m} \Delta_R^*(P_i \cdot \{a\} \cdot S_i) \cup \bigcup_{i,j=1}^{m} \Delta_R^*(P_i \cdot \{a\} \cdot K_{i,j,m} \cdot \{a\} \cdot S_j)$.

The next two claims show that also the operations of concatenation and of computing descendants mod R commute in certain instances.

Claim 3. *For all* $i \in \{1,\ldots,m\}$, $\Delta_R^*(P_i \cdot \{a\} \cdot S_i) = \Delta_R^*(P_i) \cdot \{a\} \cdot \Delta_R^*(S_i)$.

Proof. Since no rule of R contains any occurrences of the letter a, this is obvious. □

Claim 4. *For all* $i,j \in \{1,\ldots,m\}$, $\Delta_R^*(P_i \cdot \{a\} \cdot K_{i,j,m} \cdot \{a\} \cdot S_j) = \Delta_R^*(P_i) \cdot \{a\} \cdot \Delta_R^*(K_{i,j,m}) \cdot \{a\} \cdot \Delta_R^*(S_j)$.

Proof. Analogously. □

$\Delta_R^*(S_0)$, $\Delta_R^*(P_i)$, and $\Delta_R^*(S_i)$ $(i = 1,\ldots,m)$ are regular by our assumption, since S_0, P_i, S_i are regular subsets of Σ^*. It remains to prove the following result.

Claim 5. $\Delta_R^*(K_{i,j,k})$ *is a regular set for all* $i,j = 1,\ldots,m$, *and* $k = 0,1,\ldots,m$.

Proof. We proceed by induction on k, using the inductive representation of the set $K_{i,j,k}$ derived in the proof of Claim 1.

***k* = 0:** $K_{i,j,0} \subseteq \Sigma^*$.
Since $K_{i,j,0}$ is regular, $\Delta_R^*(K_{i,j,0})$ is regular by our assumption on R.

***k* = 1:** $K_{i,j,1} = K_{i,j,0} \cup K_{i,1,0} \cdot (\{a\} \cdot K_{1,1,0})^* \cdot \{a\} \cdot K_{1,j,0}$.
Hence, $\Delta_R^*(K_{i,j,1}) = \Delta_R^*(K_{i,j,0} \cup K_{i,1,0} \cdot (\{a\} \cdot K_{1,1,0})^* \cdot \{a\} \cdot K_{1,j,0}) = \Delta_R^*(K_{i,j,0}) \cup \Delta_R^*(K_{i,1,0}) \cdot (\{a\} \cdot \Delta_R^*(K_{1,1,0}))^* \cdot \{a\} \cdot \Delta_R^*(K_{1,j,0})$.
Thus, $\Delta_R^*(K_{i,j,1})$ is regular.

***k* → *k*+1:** analogously. □

Claim 6. $\Delta_R^*(S)$ *is regular.*

Proof. $\Delta_R^*(S) = \Delta_R^*(S_0) \cup \bigcup_{i=1}^m (\Delta_R^*(P_i) \cdot \{a\} \cdot \Delta_R^*(S_i)) \cup \bigcup_{i,j=1}^m (\Delta_R^*(P_i) \cdot \{a\} \cdot \Delta_R^*(K_{i,j,m}) \cdot \{a\} \cdot \Delta_R^*(S_j))$. Hence, $\Delta_R^*(S)$ is indeed a regular set. □

This completes the proof of Lemma 3.1. □

From this lemma we obtain Gyenizse's and Vágvölgyi's conjecture.

Theorem 3.2. *Let R be a string-rewriting system, and let* Σ *be the alphabet consisting of all the letters that have occurrences in R. Assume that* Σ *is finite. Then R preserves* Σ*-regularity if and only if R preserves regularity.*

Proof. Let Δ be a finite alphabet containing Σ. If $\Delta = \Sigma$, then R preserves Δ-regularity. Otherwise, let $\Delta \backslash \Sigma = \{a_1, a_2, \ldots, a_n\}$. Using Lemma 3.1 repeatedly, we see that R preserves Δ-regularity if it preserves Σ-regularity. □

Thus for string-rewriting systems, when we talk about the property of preserving regularity, there is no need to specify the underlying alphabet in detail as long as it contains all the letters that have occurrences in the rules of the string-rewriting system considered. Observe that the proof of Lemma 3.1 is constructive in that, if R preserves Σ-regularity in an effective way, then it preserves Γ-regularity in an effective way, too. That is, if a regular language $S \subseteq \Gamma^*$ is given through some finite state acceptor, then a finite state acceptor for the language $\Delta_R^*(S)$ can be constructed effectively.

If Σ is a finite alphabet, then F_Σ denotes the signature $F_\Sigma := \{a(.) \mid a \in \Sigma\} \cup \{¢\}$, where each letter $a \in \Sigma$ is interpreted as a unary function symbol $a(.)$, and ¢ is a constant. For a string-rewriting system R on $\Sigma, S_R := \{\ell(x) \to r(x) \mid (\ell \to r) \in R\}$ is the corresponding term-rewriting system on the signature F_Σ. The mapping $\alpha : \Sigma^* \to T(F_\Sigma)$ that is defined by taking $\alpha(w) := w(¢)$ induces a bijection between the regular languages on Σ and the regular tree languages over $T(F_\Sigma)$. Hence, the term-rewriting system S_R preserves F_Σ-regularity if and only if the string-rewriting system R preserves (Σ-)regularity.

Now S_R preserves regularity if it preserves F-regularity for each signature F containing F_Σ. Observe that F may contain additional constants and function symbols of arity larger than one in contrast to the case of strings considered above. Actually, S_R preserves regularity if it preserves F_1-regularity, where the signature F_1 is obtained from F_Σ by introducing a binary function symbol h and an additional constant \$ [14]. Using the same basic idea as in the proof of Lemma 3.1 the following can now be shown.

Theorem 3.3. *The term-rewriting system S_R preserves regularity if and only if it preserves F_Σ-regularity.*

Proof. If S_R preserves regularity, then is obviously also preserves F_Σ-regularity. To prove the converse implication let us assume that S_R preserves F_Σ-regularity, which means that the string-rewriting system R preserves Σ-regularity, and let $F_1 := F_\Sigma \cup \{h, \$\}$, where h is a binary function symbol and \$ is a new constant. It suffices to show that S_R preserves F_1-regularity.

To this end let $S \subset T(F_1)$ be a regular tree language. Then there exists a deterministic bottom-up tree automaton (dbuta) $A := (F_1, Q, Q_f, \delta)$ accepting S, where Q is the finite set of states, $Q_f \subset Q$ is the set of final states, and δ is a set of transitions of the following forms:

(i) $c \to q$	for $c \in \{¢, \$\}$ and some $q \in Q$,
(ii) $a(q_1) \to q$	for $a \in \Sigma$ and some $q_1, q \in Q$, and
(iii) $h(q_1, q_2) \to q$	for some $q_1, q_2, q \in Q$,

that is, $S = \{t \in T(F_1) \mid \exists q \in Q_f : t \to_A^* q\}$. Since A is deterministic, no two different transitions have the same left-hand side.

From A we construct a non-deterministic bottom-up tree automaton (nbuta) that accepts the language $\Delta_{S_R}^*(S)$. This construction proceeds in two stages.

Let $\overline{Q}$ and $\widehat{Q}$ be two sets that both are in one-to-one correspondence to the set Q. For $q \in Q$ the corresponding elements of $\overline{Q}$ and $\widehat{Q}$ will be denoted by $\bar{q}$ and $\hat{q}$, respectively. Without loss of generality we may assume that the three sets $Q, \overline{Q}$, and $\widehat{Q}$ are pairwise disjoint. Let $P := Q \cup \overline{Q} \cup \widehat{Q}$, and let $P_f := \widehat{Q}_f = \{\hat{q} \mid q \in Q_f\}$. The nbuta B is defined as $B := (F_1, P, P_f, \delta')$, where δ' is the following set of transitions:

(i) $c \to \bar{q}$	if $c \in \{¢, \$\}$ and $(c \to q) \in \delta$,
(ii) $a(q_1) \to q$	if $a \in \Sigma$ and $(a(q_1) \to q) \in \delta$,
(iii) $h(\hat{q}_1, \hat{q}_2) \to \bar{q}$	if $(h(q_1, q_2) \to q) \in \delta$, and
(iv) $\bar{q} \to q$ and $q \to \hat{q}$	for all $q \in Q$.

It is easily verified that $L(B) = L(A)$ holds, that is, B also accepts the language S. In B all h-transitions start from states in $\widehat{Q}$ and end in states from $\overline{Q}$, that is, the h-transitions have been isolated from the other transitions.

Now for $q_1, q_2 \in Q$, let $L(q_1, q_2) := \{w \in \Sigma^* \mid$ There is a sequence of transitions from $\bar{q}_1$ to $\hat{q}_2$ in B that is labelled with $w\}$. Then $L(q_1, q_2)$ is a regular language, and hence, so is the set $\Delta_R^*(L(q_1, q_2))$. Thus, there is a nondeterministic finite state acceptor (nfsa) $C(q_1, q_2) := (Q(q_1, q_2), \Sigma, \alpha(q_1, q_2), \bar{q}_1, \hat{q}_2)$ accepting this language, where we may assume without loss of generality that $\hat{q}_2$ is the unique accepting state of $C(q_1, q_2)$, that no transition of $C(q_1, q_2)$ leaves this state, and that no transition of $C(q_1, q_2)$ enters its initial state $\bar{q}_1$.

Finally, we construct an nbuta C as follows. It consists of the union of all the nfsas $C(q_1, q_2)$ for $q_1, q_2 \in Q$, where we assume for $(q_1, q_2), (q_3, q_4) \in Q \times Q$ that $C(q_1, q_2)$ and $C(q_3, q_4)$ have the initial state in common if and only if $q_1 = q_3$, that they have the final state in common if and only if $q_2 = q_4$, and that they do not have any other states in common. Further, for each h-transition $h(\hat{q}_1, \hat{q}_2) \to \bar{q}$ of B, C gets the corresponding h-transition. Finally, C also inherites the ¢- and \$-transitions from B, and it has the set of final states $\widehat{Q}_f$.

From this construction it is easily seen that $L(C)$ is a subset of $\Delta_{S_R}^*(S)$. On the other hand, we can conclude from the form of the rules of S_R that each term $t \in \Delta_{S_R}^*(S)$ is accepted by C, that is, $L(C) = \Delta_{S_R}^*(S)$. Thus, $\Delta_{S_R}^*(S)$ is indeed a regular tree language, proving that S_R is really F_1-regularity preserving.

This completes the proof of Theorem 3.3. □

As we will see in the next section, it is undecidable in general whether a finite string-rewriting system R on Σ preserves regularity (Theorem 4.2). From Theorem 3.3 and the remarks preceding it we see that the string-rewriting system R preserves regularity if and only if the corresponding term-rewriting system S_R preserves regularity. Hence, we obtain the following negative answer to a question of Gyenizse and Vágvölgyi [14].

Corollary 3.4. *It is undecidable in general whether a finite term-rewriting system preserves regularity.*

4. The property of preserving regularity is undecidable

Here we establish two undecidability results concerning the property of preserving regularity for string-rewriting systems. The first one says that it is undecidable in general whether a given string-rewriting system preserves regularity. This result is based on the following characterization.

Lemma 4.1. *Let R be a finite string-rewriting system on Σ such that the monoid M_R presented by $(\Sigma; R)$ is a group. Then the following two statements are equivalent*:
(a) *the string-rewriting system $R \cup R^{-1}$ preserves regularity,*
(b) *the group M_R is finite.*
Here R^{-1} denotes the string-rewriting system $R^{-1} := \{r \to \ell \mid (\ell \to r) \in R\}$.

Proof. (b)⇒(a): Assume that M_R is a finite group. Let $S \subseteq \Sigma^*$ be a regular language. Then $\Delta^*_{R \cup R^{-1}}(S) = [S]_R$. Since M_R is finite, there are finitely many strings $w_1, \ldots, w_n \in S$ such that $[S]_R = \bigcup_{i=1,\ldots,n} [w_i]_R$. Since M_R is a finite group, $[w]_R$ is a regular language for each $w \in \Sigma^*$. Thus, $\Delta^*_{R \cup R^{-1}}(S) = \bigcup_{i=1,\ldots,n} [w_i]_R$ is a regular language, that is, $R \cup R^{-1}$ preserves regularity.

(a)⇒(b): Assume that $R \cup R^{-1}$ preserves regularity. The singleton set $\{\lambda\} \subseteq \Sigma^*$ is a regular language, and $\Delta^*_{R \cup R^{-1}}(\lambda) = [\lambda]_R$. Since $R \cup R^{-1}$ preserves regularity, $[\lambda]_R$ is a regular language, and hence, M_R is a regular group. However, a finitely presented group is regular if and only if it is finite [1]. Hence, M_R is a finite group. □

However, the property of being finite is a Markov property of finitely presented groups, and hence, the following decision problem cannot be solved algorithmically (see, e.g., [19]):

Instance: A finite group-presentation $\langle \Sigma; L \rangle$.
Question: Is the group G_L presented by $\langle \Sigma; L \rangle$ finite?

Using Lemma 4.1 we now reduce this undecidable problem to the problem of deciding whether or not a finite string-rewriting system preserves regularity.

Theorem 4.2. *The following problem is undecidable in general*:
Instance: *A finite string-rewriting system R on Σ.*
Question: *Does R preserve regularity?*

Proof. Let $\langle \Gamma; u_1, \ldots, u_n \rangle$ be a finite group-presentation. Let $\overline{\Gamma}$ be an alphabet in one-to-one correspondence to Γ such that $\Gamma \cap \overline{\Gamma} = \emptyset$, let $\underline{\Gamma} := \Gamma \cup \overline{\Gamma}$, and let R be the string-rewriting system on $\underline{\Gamma}$ containing the following rules:

$$\begin{array}{lll} a\bar{a} \to \lambda, & \bar{a}a \to \lambda & (a \in \Gamma), \\ u_i \to \lambda & (i = 1, \ldots, n). & \end{array}$$

Then $(\underline{\Gamma}; R)$ is a finite monoid-presentation for the group G presented by $\langle \Gamma; u_1, \ldots, u_n \rangle$.

Now the string-rewriting system $R \cup R^{-1}$ preserves regularity if and only if the group G is finite (Lemma 4.1). Since $R \cup R^{-1}$ is easily obtained from the given group-presentation $\langle \Gamma; u_1, \ldots, u_n \rangle$, Theorem 4.2 follows from the undecidability result above. $\square$

Actually, the proof of Lemma 4.1 shows that the group G presented by $\langle \Gamma; u_1, \ldots, u_n \rangle$ is finite if and only if the language $\Delta^*_{R \cup R^{-1}}(\lambda)$ is regular. Thus, we actually have the following stronger undecidability result.

Corollary 4.3. *The following problem is undecidable in general:*
Instance: *A finite string-rewriting system R on Σ, and a regular language $S \subseteq \Sigma^*$.*
Question: *Is the language $\Delta^*_R(S)$ regular?*

In fact, Corollary 4.3 remains valid even if the language S is fixed to the set $S := \{\lambda\}$.

Corollary 4.4. *The following problem is undecidable in general:*
Instance: *A finite string-rewriting system R on Σ.*
Question: *Is the language $\Delta^*_R(\lambda)$ regular?*

Theorem 4.2 and Corollary 4.3 improve upon Theorem 7 and Theorem 8 of [12], since here we are only dealing with strings, i.e., with signatures containing only unary function symbols and possibly a single constant. In addition, our proof is much simpler than the one given in [12].

Our second undecidability result improves upon Corollary 4.3. It states that the problem considered in Corollary 4.3 remains undecidable even if R is restricted to finite convergent string-rewriting systems. Actually, a fixed convergent system R can be chosen here, if it is constructed accordingly. Below such a construction is described in detail.

Let $M = (Q, \Sigma, \delta, q_0, q_a)$ be a deterministic single-tape Turing machine that accepts a language $L \subseteq \Sigma^*$, where $\Sigma_b := \Sigma \cup \{b\}$ is the tape alphabet, b denotes the blank symbol, Q is the set of states, $\delta : (Q \backslash \{q_a\}) \times \Sigma_b \to Q \times (\Sigma \cup \{\ell, r\})$ is the transition function, where ℓ and r stand for the operation of moving M's head one step to the left or right, respectively, $q_0 \in Q$ is the initial state, $q_a \in Q$ is the final state, and $\vdash^*_M$ denotes the reflexive and transitive closure of the single-step computation relation $\vdash_M$ of M. Without loss of generality we may assume that $\Sigma_b \cap Q = \emptyset$.

Observe that the Turing machine considered here cannot print the blank symbol b, that is, it cannot erase the tape inscription. Actually, we can assume that the following equivalence holds for each $w \in \Sigma^*$:

$$w \in L \text{ if and only if } q_0 w \vdash^*_M u q_a v \text{ for some } u, v \in \Sigma^*.$$

From M we construct another single-tape Turing machine $\overline{M} = (\overline{Q}, \Gamma, \bar{\delta}, q_0, q_a)$. Let $\Gamma := \Sigma \cup \{1, 2, \uparrow\}$, where 1, 2, and $\uparrow$ are three new symbols, and define $\overline{Q}$ and $\bar{\delta}$ in such a way that $\overline{M}$ simulates the Turing machine M as follows:

Whenever $u_1q_1v_1 \vdash_M u_2q_2v_2$, where $u_1, v_1, u_2, v_2 \in \Sigma^*$ and $q_1, q_2 \in Q$, then $\overline{M}$ performs the following computation for each $n \geqslant 0$:

$$\begin{aligned} 1^n2^nu_1q_1v_1 &\vdash_{\overline{M}}^* \tilde{q}_1 1^n2^nu_1 \uparrow v_1 \vdash_{\overline{M}}^* 1^{n+1}2^{n+1}\hat{q}_1u_1 \uparrow v_1 \\ &\vdash_{\overline{M}}^* 1^{n+1}2^{n+1}u_1\bar{q}_1v_1 \vdash_{\overline{M}} 1^{n+1}2^{n+1}u_2q_2v_2. \end{aligned}$$

Thus, if $q_0w \vdash_M^n uqv$ for some $w, u, v \in \Sigma^*$, $q \in Q$, and $n \in \mathbb{N}$, then $q_0w \vdash_{\overline{M}}^* 1^n2^nuqv$. In particular, $\overline{M}$ also accepts the language L.

For $w \in \Sigma^*$, let $\Delta_{\overline{M}}(w) := \{uqv \mid q_0w \vdash_{\overline{M}}^* uqv \in \Gamma^* \cdot \overline{Q} \cdot \Gamma^*\}$.

Lemma 4.5. *For $w \in \Sigma^*$, the following two statements are equivalent*:
(a) *$\Delta_{\overline{M}}(w)$ is a regular language*;
(b) *$w \in L$.*

Proof. (b)$\Rightarrow$(a): If $w \in L$, then $\overline{M}$ accepts on input w, that is, $\overline{M}$ halts on input w after performing a finite number of steps. Thus, the set $\Delta_{\overline{M}}(w)$ is finite.

(a)$\Rightarrow$(b): If $w \notin L$, then $\overline{M}$ does not halt on input w, and the same is true for M. Thus, there is an infinite computation of the form

$$q_0w \vdash_M u_1q_1v_1 \vdash_M u_2q_2v_2 \vdash_M \ldots \vdash_M u_iq_iv_i \vdash_M u_{i+1}q_{i+1}v_{i+1} \vdash_M \ldots.$$

Hence, $\overline{M}$ performs an infinite computation of the following form:

$$\begin{aligned} q_0w \vdash_{\overline{M}}^* 12u_1q_1v_1 \vdash_{\overline{M}}^* 1^22^2u_2q_2v_2 \vdash_{\overline{M}}^* \ldots \vdash_{\overline{M}}^* 1^i2^iu_iq_iv_i \\ \vdash_{\overline{M}}^* 1^{i+1}2^{i+1}u_{i+1}q_{i+1}v_{i+1} \vdash_{\overline{M}}^* \ldots. \end{aligned}$$

Assume that the set $\Delta_{\overline{M}}(w)$ were regular. Consider the set

$$\Delta'(w) := \Delta_{\overline{M}}(w) \cap 1^+ \cdot 2^+ \cdot ((\Gamma \backslash \{1,2\}) \cup \overline{Q})^+.$$

With $\Delta_{\overline{M}}(w)$ also the set $\Delta'(w)$ is regular. However, if $z \in \Delta'(w)$, then there exist $i \in \mathbb{N}_+$ and $z' \in ((\Gamma\backslash\{1,2\}) \cup \overline{Q})^+$ such that $z = 1^i2^iz'$, and for each $i \in \mathbb{N}_+$, $1^i2^iz' \in \Delta'(w)$ for some $z' \in ((\Gamma\backslash\{1,2\}) \cup \overline{Q})^+$. Thus, the set $\Delta'(w)$ does not satisfy the pumping lemma for regular languages, and hence, $\Delta'(w)$ is not regular. Accordingly, the set $\Delta_{\overline{M}}(w)$ is not regular, either. $\square$

From the Turing machine $\overline{M}$ we now construct a finite, length-reducing, and confluent string-rewriting system $R(\overline{M})$ that simulates the computations of $\overline{M}$. Let \$, ¢, and d be three additional symbols, and let $\Gamma_0 := \Gamma_b \cup \overline{Q} \cup \{\$, ¢, d\}$. The symbols \$ and ¢ will serve as left and right end markers, respectively, of encodings of configurations of $\overline{M}$, while the symbol d is being used to ensure that the rules of $R(\overline{M})$ are length-reducing. The system $R(\overline{M})$ consists of the following three groups of rules:

(1) Rules to simulate the stepwise behaviour of $\overline{M}$:

$$\begin{array}{rcll}
q_i a_k dd & \to & q_j a_\ell & \text{if } \bar{\delta}(q_i, a_k) = (q_j, a_\ell) \\
q_i ¢ dd & \to & q_j a_\ell ¢ & \text{if } \bar{\delta}(q_i, b) = (q_j, a_\ell) \\
q_i a_k dd & \to & a_k q_j & \text{if } \bar{\delta}(q_i, a_k) = (q_j, r) \\
q_i ¢ dd & \to & b q_j ¢ & \text{if } \bar{\delta}(q_i, b) = (q_j, r) \\
a_\ell q_i a_k dd & \to & q_j a_\ell a_k & \text{if } \bar{\delta}(q_i, a_k) = (q_j, \ell) \\
a_\ell q_i ¢ dd & \to & q_j a_\ell ¢ & \text{if } \bar{\delta}(q_i, b) = (q_j, \ell) \\
\$ q_i a_k dd & \to & \$ q_j b a_k & \text{if } \bar{\delta}(q_i, a_k) = (q_j, \ell) \\
\$ q_i ¢ dd & \to & \$ q_j b ¢ & \text{if } \bar{\delta}(q_i, b) = (q_j, \ell)
\end{array}
\left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\right\} a_\ell \in \Gamma_b$$

(2) Rules to shift occurrences of the symbol d to the left:

$$\left.\begin{array}{rcl}
a_i a_j dd & \to & a_i d a_j \\
a_i d a_j dd & \to & a_i dd a_j
\end{array}\right\} \text{for all } a_i \in \Gamma_b, a_j \in \Gamma_b \cup \{¢\}$$

(3) Rules to erase halting configurations:

$$\left.\begin{array}{rcl}
q_a a_i dd & \to & q_a \\
a_i q_a ¢ d & \to & q_a ¢
\end{array}\right\} \text{for all } a_i \in \Gamma_b$$

$$\$ q_a ¢ d \to \$ q_a ¢$$

The system $R(\overline{M})$ has the following properties.

Proposition 4.6 (cf. [23], Proposition 3.1).
(a) *The string-rewriting system $R(\overline{M})$ is finite, length-reducing, and confluent.*
(b) *For $w \in \Sigma^*$, the following two statements are equivalent:*
 (1) $w \in L$; *and*
 (2) $\exists m \in \mathbb{N}\, \forall n \geqslant m : \$ q_0 w ¢ d^n \to^*_{R(\overline{M})} \$ q_a ¢ d^{n-m} \to^*_{R(\overline{M})} \$ q_a ¢$.

From Lemma 4.5 and Proposition 4.6(b) we obtain the following characterization.

Lemma 4.7. *For $w \in \Sigma^*$, the following two statements are equivalent:*
(a) $\Delta^*_{R(\overline{M})}(\$ q_0 w ¢ \cdot d^*)$ *is a regular language;*
(b) $w \in L$.

Proof. (b)$\Rightarrow$(a): If $w \in L$, then by Proposition 4.6(b) there exists an integer $m \in \mathbb{N}$ such that

$$\$ q_0 w ¢ d^n \to^*_{R(\overline{M})} \$ q_a ¢ d^{n-m} \to^{n-m}_{R(\overline{M})} \$ q_a ¢$$

holds for all $n \geqslant m$. Thus,

$$\Delta^*_{R(\overline{M})}(\$q_0w¢ \cdot d^*) = \bigcup_{i=0}^{m} \Delta^*_{R(\overline{M})}(\$q_0w¢ \cdot d^i) \cup \Delta^*_{R(\overline{M})}(\$q_0w¢ \cdot d^m) \cdot d^*.$$

Since $R(\overline{M})$ is length-reducing, each of the finitely many languages $\Delta^*_{R(\overline{M})}(\$q_0w¢ \cdot d^i), i = 0, 1, \ldots, m$, is finite. Thus, $\Delta^*_{R(\overline{M})}(\$q_0w¢ \cdot d^*)$ is a regular language.

(a)⇒(b): Assume that $w \notin L$. Let $\varphi : \Gamma_0 \to \Gamma_b \cup \overline{Q}$ denote the projection, that is, $\varphi(a) := a$ for all $a \in \Gamma_b \cup \overline{Q}$, and $\varphi(\$) = \varphi(¢) = \varphi(d) = \lambda$. Then $\varphi(\Delta^*_{R(\overline{M})}(\$q_0w¢ \cdot d^*)) = \Delta_{\overline{M}}(w)$, which is not regular by Lemma 4.5. Hence, the set $\Delta^*_{R(\overline{M})}(\$q_0w¢ \cdot d^*)$ is not regular, either. □

Now choose L to be a nonrecursive language. Then Lemma 4.7 yields the following undecidability result.

Theorem 4.8. *There exists a finite, length-reducing, and confluent string-rewriting system R such that the following problem is undecidable*:
Instance: *A regular set* $S \subseteq \Sigma^*$.
Question: *Is the set of descendants* $\Delta^*_R(S)$ *regular*?

If $w \in L$, then $\Delta^*_{R(\overline{M})}(\$q_0w¢ \cdot d^n) \cap \mathrm{IRR}(R(\overline{M})) = \{\$q_a¢\}$ for all $n \geqslant m$. Thus,

$$\mathrm{NF}_{R(\overline{M})}(\$q_0w¢ \cdot d^*) = \Delta^*_{R(\overline{M})}(\$q_0w¢ \cdot d^*) \cap \mathrm{IRR}(R(\overline{M}))$$

is a finite set, and hence, it is regular. On the other hand, if $w \notin L$, then

$$\varphi(\Delta^*_{R(\overline{M})}(\$q_0w¢ \cdot d^*) \cap \mathrm{IRR}(R(\overline{M}))) = \Delta_{\overline{M}}(w),$$

which is not a regular set in this case by Lemma 4.5. Thus,

$$\mathrm{NF}_{R(\overline{M})}(\$q_0w¢ \cdot d^*) = \Delta^*_{R(\overline{M})}(\$q_0w¢ \cdot d^*) \cap \mathrm{IRR}(R(\overline{M}))$$

is not regular, if $w \notin L$, that is, the set of normal forms in the language $\Delta^*_{R(\overline{M})}(\$q_0w¢ \cdot d^*)$ is regular if and only if $w \in L$. Thus, we obtain the following corollary, which improves upon Theorem 10 of [12].

Corollary 4.9. *There exists a finite, length-reducing, and confluent string-rewriting system R such that the following problem is undecidable*:
Instance: *A regular set* $S \subseteq \Sigma^*$.
Question: *Is the set* $\mathrm{NF}_R(S)$ *regular*?

For the string-rewriting system $R(\overline{M})$ and the languages of the form $\$q_0w¢ \cdot d^* (w \in \Sigma^*)$, the following properties hold:
If $w \in L$, then $\Delta^*_{R(\overline{M})}(\$q_0w¢ \cdot d^*)$ is a regular language, and hence, also

$$\mathrm{NF}_{R(\overline{M})}(\$q_0w¢ \cdot d^*) = \Delta^*_{R(\overline{M})}(\$q_0w¢ \cdot d^*) \cap \mathrm{IRR}(R(\overline{M}))$$

is a regular language, but if $w \notin L$, then neither $\Delta^*_{R(\overline{M})}(\$q_0w¢ \cdot d^*)$ nor $\mathrm{NF}_{R(\overline{M})}$ $(\$q_0w¢ \cdot d^*)$ is a regular language. Thus, $\mathrm{NF}_{R(\overline{M})}(\$q_0w¢ \cdot d^*)$ is regular if and only if $\Delta^*_{R(\overline{M})}(\$q_0w¢ \cdot d^*)$ is regular. However, by adding some rules to the string-rewriting system $R(\overline{M})$ we obtain a string-rewriting system $R_0(\overline{M})$ on $\Gamma_0 \cup \{\#, z\}$, which does not satisfy this equivalence. Here # and z are two new symbols. Thus, for some regular languages S, $\mathrm{NF}_{R_0(\overline{M})}(S)$ will be regular even if $\Delta^*_{R_0(\overline{M})}(S)$ is not.

Let $\Gamma_1 := \Gamma_0 \cup \{\#, z\}$, and let $R_0(\overline{M}) := R(\overline{M}) \cup R_0$, where R_0 contains the following rules:

$$¢\# \to z,\ \ ¢d\# \to z,\ \ za \to z,\ \ az \to z \quad (a \in \Gamma_1).$$

Then $R_0(\overline{M})$ is a finite length-reducing string-rewriting system, and it can easily be verified that $R_0(\overline{M})$ is confluent.

For $w \in \Sigma^*$, consider the language $S(w) := \$q_0w¢ \cdot d^* \cdot \#$. If $w \in L$, and if n is sufficiently large, then

$$\$q_0w¢ \cdot d^n \cdot \# \to^*_{R(\overline{M})} \$q_a¢\# \to^*_{R(\overline{M})} z.$$

If n is small, then

$$\$q_0w¢ \cdot d^n \cdot \# \to^*_{R(\overline{M})} \$uqv¢d^\varepsilon \cdot \# \to^*_{R(\overline{M})} z$$

for some $\varepsilon \in \{0,1\}$. Hence, $\Delta^*_{R_0(\overline{M})}(S(w))$ is a regular language, and $\mathrm{NF}_{R_0(\overline{M})}(S(w)) = \{z\}$. If $w \notin L$, then, for all $n \in \mathbb{N}$,

$$\$q_0w¢ \cdot d^n \cdot \# \to^*_{R(\overline{M})} \$uqv¢d^\varepsilon \cdot \# \to^*_{R(\overline{M})} z,$$

where $\varepsilon \in \{0,1\}$. The rules of R_0 cannot be used before all the d's (but one) to the right of the ¢-symbol have been used up. Thus, it follows as in the proof of Lemma 4.7 that $\Delta^*_{R_0(\overline{M})}(S(w))$ is not regular. However, $\mathrm{NF}_{R_0(\overline{M})}(S(w)) = \{z\}$ holds also in this situation. Thus, the following undecidability result follows.

Theorem 4.10. *There exists a finite, length-reducing, and confluent string-rewriting system R such that the following problem is undecidable:*
Instance: A regular set $S \subseteq \Sigma^*$ such that $\mathrm{NF}_R(S)$ is a singleton.
Question: Is the set of descendants $\Delta^*_R(S)$ regular?

So far we have seen that Problems 2 and 4 are undecidable, even for a fixed finite string-rewriting system that is length-reducing and confluent (Theorem 4.8 and Corollary 4.9). In the remaining part of the paper we want to prove that also Problem 3 remains undecidable in general when it is restricted to finite string-rewriting systems that are length-reducing and confluent.

In principle the proof will be similar to the above proof of Theorem 4.8. Given a Turing machine M, a finite, length-reducing, and confluent string-rewriting system R_M will be constructed that simulates the computations of M. If the Turing machine M

has some infinite computation, then it is easily seen that there exists a regular language S such that the language $\Delta^*_{R_M}(S)$ is not regular. Hence, in this case the system R_M is not regularity preserving. However, it seems to be very difficult to prove the converse implication, that is, even if all computations of M are finite, it is not at all clear how to guarantee that the system R_M preserves regularity.

To get around this difficulty we will make use of the additional assumption that the Turing machine M considered is strongly bounded. In order to justify this assumption we consider the strong boundedness problem for single-tape Turing machines in the next section.

5. The strong boundedness problem for Turing machines

First it should be stressed that in the following we will only be dealing with single-tape Turing machines that are deterministic. A possibly infinite configuration C of a Turing machine M is called *immortal* if M does never halt when starting from C. In [15] Hooper shows that it is undecidable whether a Turing machine has an immortal configuration. Actually, Hooper only considers *2-symbol* Turing machines, that is, Turing machines that only have a single tape symbol in addition to the blank symbol.

We call a Turing machine M *strongly bounded* if there exists an integer k such that, for each finite configuration C, M halts after at most k steps when starting from this configuration. Here a configuration is called finite if almost all tape squares contain the blank symbol b. We are interested in the *strong boundedness problem* for Turing machines, which is the following decision problem:

Instance: A single-tape Turing machine M.
Question: Is M strongly bounded?

In [15] Hooper proceeds as follows. He first observes that the halting problem is undecidable for the class of two-counter Minsky machines that start with empty counters. Then he constructs a 2-symbol Turing machine $\overline{M}$ from a Minsky machine $\hat{M}$ such that $\overline{M}$ has an immortal configuration if and only if $\hat{M}$ does not halt from its initial configuration with empty counters. Since the configurations of $\hat{M}$ are encoded as certain finite configurations of $\overline{M}$, and since $\overline{M}$ simulates $\hat{M}$, though in a very involved way, this shows that $\overline{M}$ has an immortal finite configuration if and only if it has an immortal configuration. It follows that the immortality problem is undecidable for 2-symbol Turing machines, even when it is restricted to finite configurations.

Now assume that the Turing machine $\overline{M}$ has finite computations of arbitrary length. Then it must also have an infinite computation, though one that possibly starts with an infinite configuration (cf. the proof of Corollary 6 of [17]). But then we see from the discussion above that $\overline{M}$ also has an infinite computation that starts with a finite configuration. Thus, if $\overline{M}$ has no immortal finite configuration, then it is strongly bounded. Since the converse is obvious, we obtain the following undecidability result.

Proposition 5.1. *The strong boundedness problem is undecidable for 2-symbol single-tape Turing machines.*

6. The reduction

We will prove that Problem 3 is undecidable for finite string-rewriting systems that are length-reducing and confluent by a reduction from the strong boundedness problem for Turing machines. For that we use a simulation of Turing machines through finite, length-reducing, and confluent string-rewriting systems that is based on the simple simulation given in Section 4.

Let $M = (Q, \Sigma, q_0, q_a, \delta)$ be a deterministic single-tape Turing machine, where we assume that Σ consists of the symbol a and the blank symbol b only. From M we now construct a finite string-rewriting system R for simulating M.

Let $\overline{\Sigma} := \{\bar{a}, \bar{b}\}$, let $\overline{Q}$ be another new alphabet that is in one-to-one correspondence to Q, and let $\Gamma := Q \cup \overline{Q} \cup \Sigma \cup \overline{\Sigma} \cup \{1, 2, \$, ¢, d, \bar{d}, \hat{d}, d_0, \hat{c}, \bar{c}, 0\}$, where $1, 2, \$, ¢, d, \bar{d}, \hat{d}, d_0, \hat{c}, \bar{c}, 0$ are 11 additional new symbols.

The string-rewriting system R will consist of two main parts, that is, $R := R_1 \cup R_2$, where R_1 is a system that simulates the computations of the Turing machine M step by step, and R_2 is a system that destroys unwanted strings. We first define the system R_1. It consists of the following 5 groups of rules.

(1) Rules to simulate the Turing machine M:

$$\begin{array}{lll}
q_i a_k d d a_r & \to \bar{q}_j a_\ell a_r & \text{for all } a_r \in \Sigma \cup \{¢\}, \text{if } \delta(q_i, a_k) = (q_j, a_\ell) \\
q_i ¢ d_0 d_0 & \to \bar{q}_j a_\ell ¢ & \text{if } \delta(q_i, b) = (q_j, a_\ell) \\
q_i a_k d d a_r & \to \bar{a}_k \bar{q}_j a_r & \text{for all } a_r \in \Sigma \cup \{¢\}, \text{if } \delta(q_i, a_k) = (q_j, r) \\
q_i ¢ d_0 d_0 & \to \bar{b} \bar{q}_j ¢ & \text{if } \delta(q_i, b) = (q_j, r) \\
\bar{a}_\ell q_i a_k d d a_r & \to \bar{q}_j a_\ell a_k a_r & \text{for all } a_r \in \Sigma \cup \{¢\}, \text{if } \delta(q_i, a_k) = (q_j, \ell) \\
\bar{a}_\ell q_i ¢ d_0 d_0 & \to \bar{q}_j a_\ell ¢ & \text{if } \delta(q_i, b) = (q_j, \ell)
\end{array} \left.\vphantom{\begin{array}{l} a \\ a \end{array}}\right\} \text{for all } \bar{a}_\ell \in \overline{\Sigma}$$

$$\begin{array}{lll}
\$ q_i a_k d d a_r & \to \$ \bar{q}_j b a_k a_r & \text{for all } a_r \in \Sigma \cup \{¢\}, \text{if } \delta(q_i, a_k) = (q_j, \ell) \\
\$ q_i ¢ d_0 d_0 & \to \$ \bar{q}_j b ¢ & \text{if } \delta(q_i, b) = (q_j, \ell).
\end{array}$$

(2) Rules to shift d to the left:

$$\left.\begin{array}{ll}
a_i a_j d d a_r & \to a_i d a_j a_r \\
a_i d a_j d d a_r & \to a_i d d a_j a_r
\end{array}\right\} \text{for all } a_i \in \Sigma \cup \overline{Q}, a_j \in \Sigma, \text{ and } a_r \in \Sigma \cup \{¢\}$$

$$\left.\begin{array}{ll}
a_i ¢ d_0 d_0 & \to a_i d ¢ \\
a_i d ¢ d_0 d_0 & \to a_i d d ¢
\end{array}\right\} \text{for all } a_i \in \Sigma \cup \overline{Q}$$

$$\left.\begin{array}{lll} \bar{a}_i\bar{q}_j d d a_r & \to & \bar{a}_i\bar{d}\bar{q}_j a_r \\ \bar{a}_i\bar{d}\bar{q}_j d d a_r & \to & \bar{a}_i\bar{d}\bar{d}\bar{q}_j a_r \end{array}\right\} \text{for all } \bar{a}_i \in \overline{\Sigma} \cup \{\$\}, \bar{q}_j \in \overline{Q}, \text{ and } a_r \in \Sigma \cup \{\text{¢}\}$$

(3) Rules to shift $\bar{d}$ and $\hat{d}$ to the left:

$$\left.\begin{array}{lll} \bar{a}_i\bar{a}_j\bar{d}\bar{d}\bar{a}_r & \to & \bar{a}_i\bar{d}\bar{a}_j\bar{a}_r \\ \bar{a}_i\bar{d}\bar{a}_j\bar{d}\bar{d}\bar{a}_r & \to & \bar{a}_i\bar{d}\bar{d}\bar{a}_j\bar{a}_r \end{array}\right\} \text{for all } \bar{a}_i \in \overline{\Sigma} \cup \{\$\}, \bar{a}_j \in \overline{\Sigma}, \text{ and } \bar{a}_r \in \overline{\Sigma} \cup \overline{Q} \cup Q$$

$$\left.\begin{array}{lll} 2\$\bar{d}\bar{d}\bar{a}_r & \to & 2\hat{d}\$\bar{a}_r \\ 2\hat{d}\$\bar{d}\bar{d}\bar{a}_r & \to & 2\hat{d}\hat{d}\$\bar{a}_r \end{array}\right\} \text{for all } \bar{a}_r \in \overline{\Sigma} \cup \overline{Q} \cup Q$$

$$\begin{array}{lll} 22\hat{d}\hat{d}\$ & \to & 2\hat{d}2\$ \\ 22\hat{d}\hat{d}2 & \to & 2\hat{d}22 \\ 2\hat{d}2\hat{d}\hat{d}\$ & \to & 2\hat{d}\hat{d}2\$ \\ 2\hat{d}2\hat{d}\hat{d}2 & \to & 2\hat{d}\hat{d}22 \end{array}$$

(4) Rules to increase the number of 1's and 2's:

$$\left.\begin{array}{lll} 12\hat{d}\hat{d}a & \to & 1\hat{d}2a \\ 1\hat{d}2\hat{d}\hat{d}a & \to & 11\hat{c}2a \\ 1\hat{c}2\hat{d}\hat{d}a & \to & 122\hat{c}a \end{array}\right\} \text{for all } a \in \{2, \$\}$$

(5) Rules to shift $\hat{c}$ and $\bar{c}$ to the right:

$$2\hat{c}2\hat{d}\hat{d}a \to 22\hat{c}a \qquad \text{for all } a \in \{2, \$\}$$

$$\left.\begin{array}{lll} 2\hat{c}\$\bar{d}\bar{d}\bar{a} & \to & 2\$\bar{c}\bar{a} \\ \bar{a}_i\bar{c}\bar{a}_j\bar{d}\bar{d}\bar{a} & \to & \bar{a}_i\bar{a}_j\bar{c}\bar{a} \end{array}\right\} \text{for all } \bar{a} \in \overline{\Sigma} \cup \overline{Q} \cup Q, \bar{a}_i \in \overline{\Sigma} \cup \{\$\}, \text{ and } \bar{a}_j \in \overline{\Sigma}$$

$$\bar{a}_i\bar{c}\bar{q}_j d d a \to \bar{a}_i q_j a \qquad \text{for all } \bar{a}_i \in \overline{\Sigma} \cup \{\$\}, \bar{q}_j \in \overline{Q}, \text{ and } a \in \Sigma \cup \{\text{¢}\}.$$

Obviously, R_1 is a finite and length-reducing system. Since the Turing machine M is deterministic, there are no overlaps between the rules of group (1). There are overlaps between the rules of group (1) and the rules of the groups (2) to (5), but they all resolve trivially. Also all the overlaps between the rules of groups (2) to (5) resolve trivially. Thus, R_1 is in addition confluent.

The rules of group (1) of R_1 simulate the stepwise computation of the Turing machine M. The auxiliary symbols $d_0, d, \bar{d}$, and $\hat{d}$ ensure that R_1 is length-reducing, and the rules of (2) and (3) shift occurrences of these auxiliary symbols to the left. After simulating a step of M, the prefix $1^{\ell}2^k$ of the encoding of the actual configuration of M is incremented to $1^{\ell+1}2^{k+1}$ through the rules of group (4). Finally, the auxiliary symbols $\hat{c}$ and $\bar{c}$, and the copies $\overline{Q}$ of the actual state symbols Q are used to ensure that the next step of M can be simulated only after the prefix $1^{\ell}2^{k}$ has been incremented (see (5)). The following technical lemma describes the behaviour of R_1 in more detail.

Lemmma 6.1.
(i) $\forall n \geqslant 1\ \forall a_1 \in \Sigma \cup \overline{Q}\ \forall a_2,\dots,a_n \in \Sigma: \quad a_1a_2\dots a_n ¢ d_0^{2^{n+1}} \to_{R_1}^{*} a_1d^2a_2\dots a_n¢.$

(ii) $\forall m,n \geqslant 0\ \forall \bar a_0 \in \overline{\Sigma} \cup \{\$\}\ \forall \bar a_1,\dots,\bar a_m \in \overline{\Sigma}\ \forall \bar q \in \overline{Q}\ \forall a'_1,\dots,a'_n \in \Sigma:$
$$\bar a_0\bar a_1\dots\bar a_m\bar q a'_1\dots a'_n¢ d_0^{2^{m+n+3}} \to_{R_1}^{*} \bar a_0\bar d^2\bar a_1\dots\bar a_m\bar q a'_1\dots a'_n¢.$$

(iii) $\forall t,m,n \geqslant 0\ \forall \bar a_1,\dots,\bar a_m \in \overline{\Sigma}\ \forall \bar q \in \overline{Q}\ \forall a'_1,\dots,a'_n \in \Sigma:$
$$2^{t+1}\$\bar a_1\dots\bar a_m\bar q a'_1\dots a'_n¢ d_0^{2^{t+m+n+4}} \to_{R_1}^{*} 2\hat d^2 2^t\$\bar a_1\dots\bar a_m\bar q a'_1\dots a'_n¢.$$

(iv) $\forall t \geqslant 1\ \forall m,n \geqslant 0\ \forall \bar a_1,\dots,\bar a_m \in \overline{\Sigma}\ \forall \bar q \in \overline{Q}\ \forall a'_1,\dots,a'_n \in \Sigma:$
$$12^t\$\bar a_1\dots\bar a_m\bar q a'_1\dots a'_n¢ d_0^{2^{t+m+n+3}} \to_{R_1}^{*} 1\hat d 2^t\$\bar a_1\dots\bar a_m\bar q a'_1\dots a'_n¢.$$

(v) $\forall s,t \geqslant 1\ \forall m,n \geqslant 0\ \forall \bar a_1,\dots,\bar a_m \in \overline{\Sigma}\ \forall \bar q \in \overline{Q}\ \forall a'_1,\dots,a'_n \in \Sigma:$
$$1^s2^t\$\bar a_1\dots\bar a_m\bar q a'_1\dots a'_n¢ d_0^{2^{t+m+n+5}-2^{m+n+4}} \to_{R_1}^{*} 1^{s+1}2^{t+1}\hat c\$\bar a_1\dots\bar a_m\bar q a'_1\dots a'_n¢.$$

(vi) $\forall m,n \geqslant 0\ \forall \bar a_1,\dots,\bar a_m \in \overline{\Sigma}\ \forall \bar q \in \overline{Q}\ \forall a'_1,\dots,a'_n \in \Sigma:$
$$2\hat c\$\bar a_1\dots\bar a_m\bar q a'_1\dots a'_n¢ d_0^{2^{m+n+4}-2^{n+2}} \to_{R_1}^{*} 2\$\bar a_1\dots\bar a_m q a'_1\dots a'_n¢.$$

Proof. (i) We proceed by induction on n:

$\boldsymbol{n = 1}$: $\quad a_1¢d_0^4 \to_{R_1} a_1d¢d_0^2 \to_{R_1} a_1dd¢.$

$\boldsymbol{n \to n+1}$: $a_1a_2\dots a_{n+1}¢d_0^{2^{n+2}}$

$\to_{R_1}^{*} \underline{a_1a_2d^2a_3}\dots a_{n+1}¢d_0^{2^{n+1}}$ (by the induction hypothesis)

$\to_{R_1} a_1da_2a_3\dots a_{n+1}¢d_0^{2^{n+1}}$

$\to_{R_1}^{*} \underline{a_1da_2d^2a_3}\dots a_{n+1}¢$ (by the induction hypothesis)

$\to_{R_1} a_1d^2a_2a_3\dots a_{n+1}¢.$

(ii) We proceed by induction on m:

$\boldsymbol{m = 0}$: $\bar a_0\bar q a'_1\dots a'_n¢d_0^{2^{n+3}} \to_{R_1}^{*} \underline{\bar a_0\bar q d^2 a'_1}\dots a'_n¢d_0^{2^{n+2}}$ (by (i))

$\to_{R_1} \bar a_0\bar d\bar q a'_1\dots a'_n¢d_0^{2^{n+2}} \to_{R_1}^{*} \underline{\bar a_0\bar d\bar q d^2 a'_1}\dots a'_n¢$ (by (i))

$\to_{R_1} \bar a_0\bar d^2\bar q a'_1\dots a'_n¢.$

$\boldsymbol{m \to m+1}$: $\bar a_0\bar a_1\dots\bar a_{m+1}\bar q a'_1\dots a'_n¢d_0^{2^{m+n+4}}$

$\to_{R_1}^{*} \underline{\bar a_0\bar a_1\bar d^2\bar a_2}\dots\bar a_{m+1}\bar q a'_1\dots a'_n¢d_0^{2^{m+n+3}}$ (by the induction hypothesis)

$\to_{R_1} \bar a_0\bar d\bar a_1\bar a_2\dots\bar a_{m+1}\bar q a'_1\dots a'_n¢d_0^{2^{m+n+3}}$

$\to_{R_1}^{*} \underline{\bar a_0\bar d\bar a_1\bar d^2\bar a_2}\dots\bar a_{m+1}\bar q a'_1\dots a'_n¢$ (by the induction hypothesis)

$\to_{R_1} \bar a_0\bar d^2\bar a_1\bar a_2\dots\bar a_{m+1}\bar q a'_1\dots a'_n¢.$

(iii) We proceed by induction on t:

$$t = 0\text{: } 2\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢d_0^{2^{m+n+4}} \to^*_{R_1} \underline{2\$\bar{d}^2\bar{a}_1} \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢d_0^{2^{m+n+3}} \quad \text{(by (ii))}$$

$$\to_{R_1} 2\hat{d}\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢d_0^{2^{m+n+3}} \to^*_{R_1} \underline{2\hat{d}\$\bar{d}^2\bar{a}_1} \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢ \quad \text{(by (ii))}$$

$$\to_{R_1} 2\hat{d}^2\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢.$$

$$t \to t+1\text{: } 2^{t+2}\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢d_0^{2^{t+m+n+5}}$$

$$\to^*_{R_1} \underline{22\hat{d}^2 2^t}\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢d_0^{2^{t+m+n+4}} \quad \text{(by the induction hypothesis)}$$

$$\to_{R_1} 2\hat{d}2^{t+1}\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢d_0^{2^{t+m+n+4}}$$

$$\to^*_{R_1} \underline{2\hat{d}2\hat{d}^2 2^t}\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢ \quad \text{(by the induction hypothesis)}$$

$$\to_{R_1} 2\hat{d}^2 2^{t+1}\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢.$$

(iv) By (iii) we have $12^t\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢d_0^{2^{t+m+n+3}} \to^*_{R_1} \underline{12\hat{d}^2 2^{t-1}}\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢ \to_{R_1} 1\hat{d}2^t\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢$.

(v) $1^s2^t\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢d_0^{2^{t+m+n+5}-2^{m+n+4}}$

$$\to^*_{R_1} \underline{1^s\hat{d}2\hat{d}^2 2^{t-1}}\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢d_0^{2^{t+m+n+4}-2^{m+n+4}} \quad \text{(by (iii) and (iv))}$$

$$\to_{R_1} 1^{s+1}\hat{c}2^t\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢d_0^{2^{t+m+n+4}-2^{m+n+4}}$$

$$\to^*_{R_1} 1^{s+1}\hat{c}(2\hat{d}^2)^t\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢ \quad \text{(by (iii))}$$

$$\to^*_{R_1} 1^{s+1}2^{t+1}\hat{c}\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢.$$

(vi) $2\hat{c}\$\bar{a}_1 \dots \bar{a}_m\bar{q}a'_1 \dots a'_n¢d_0^{2^{m+n+4}-2^{n+2}} \to^*_{R_1} 2\hat{c}\$\bar{d}^2\bar{a}_1 \dots \bar{d}^2\bar{a}_m\bar{d}^2\bar{q}a'_1 \dots a'_n¢d_0^{2^{n+2}}$ (by (ii))

$$\to^*_{R_1} 2\$\bar{a}_1 \dots \bar{a}_m\hat{c}\bar{q}a'_1 \dots a'_n¢d_0^{2^{n+2}} \to^*_{R_1} 2\$\bar{a}_1 \dots \underline{\bar{a}_m\hat{c}\bar{q}d^2a'_1} \dots a'_n¢ \quad \text{(by (i))}$$

$$\to_{R_1} 2\$\bar{a}_1 \dots \bar{a}_mqa'_1 \dots a'_n¢.$$

This completes the proof of Lemma 6.1. □

Using this technical lemma the following result is easily derived. Here $^- : \Sigma^* \to \overline{\Sigma}^*$ denotes the canonical bijection.

Lemma 6.2. *Let uqv be a configuration of the Turing machine M such that $uqv \vdash_M u_1q_1v_1$, and let $k \geqslant 1$. Then there exists a positive integer $p \in \mathbb{N}$ such that $1^\ell 2^k\$\bar{u}qv¢d_0^p \to^*_{R_1} 1^{\ell+1}2^{k+1}\$\bar{u}_1q_1v_1¢$ holds for all $\ell \geqslant 1$.*

Proof. Let $\ell, k \geqslant 1$, and let $uqv = a_1 \dots a_mqa'_1 \dots a'_n \vdash_M a_1 \dots a_mq_1\tilde{a}_1a'_2 \dots a'_n = u_1q_1v_1$, that is, $\delta(q, a'_1) = (q_1, \tilde{a}_1)$. The other cases can be dealt with analogously.

(1) $1^{\ell}2^{k}\$\bar{u}qv¢d_0^{2^{n+1}} \to_{R_1}^{*} 1^{\ell}2^{k}\$\bar{u}\underline{qa_1'd^2a_2'}\dots a_n'¢$ (by Lemma 6.1(i))

$\to_{R_1} 1^{\ell}2^{k}\$\bar{u}\bar{q}_1\tilde{a}_1a_2'\dots a_n'¢$.

(2) $1^{\ell}2^{k}\$\bar{u}\bar{q}_1\tilde{a}_1a_2'\dots a_n'¢d_0^{2^{k+m+n+5}-2^{n+2}}$

$\to_{R_1}^{*} 1^{\ell+1}2^{k+1}\hat{c}\$\bar{u}\bar{q}_1\tilde{a}_1a_2'\dots a_n'¢\cdot d_0^{2^{m+n+4}-2^{n+2}}$ (by Lemma 6.1(v))

$\to_{R_1}^{*} 1^{\ell+1}2^{k+1}\$\bar{u}q_1\tilde{a}_1a_2'\dots a_n'¢$ (by Lemma 6.1(vi))

$= 1^{\ell+1}2^{k+1}\$\bar{u}_1q_1v_1¢$.

Hence, for this case we obtain $p = 2^{k+m+n+5} - 2^{n+1}$. □

This result has the following consequence.

Lemma 6.3. *If the Turing machine M has an immortal finite configuration, then R_1 does not preserve regularity.*

Proof. Assume that $u_0q_0v_0$ is an immortal finite configuration of M, that is, there exists an infinite computation of M of the form

$$u_0q_0v_0 \vdash_M u_1q_1v_1 \vdash_M u_2q_2v_2 \vdash_M \cdots \vdash_M u_iq_iv_i \vdash_M \cdots.$$

Consider the regular language $S := \{12\$\bar{u}_0q_0v_0¢\cdot d_0^i \mid i \geqslant 0\}$. From Lemma 6.2 we see that, for each $k \geqslant 1$, there exists an integer $p_k \in \mathbb{N}_+$ such that $12\$\bar{u}_0q_0v_0¢\cdot d_0^{p_k} \to_{R_1}^{*} 1^{k+1}2^{k+1}\$\bar{u}_kq_kv_k¢$. Hence, we see from the form of the rules of R_1 that

$$\Delta_{R_1}^{*}(S) \cap 1^{+}\cdot 2^{+}\cdot \$\cdot \overline{\Sigma}^{*}\cdot Q\cdot \Sigma^{*}\cdot ¢ = \{1^{k+1}2^{k+1}\$\bar{u}_kq_kv_k¢ \mid k \geqslant 0\}.$$

Since this language does not satisfy the pumping lemma for regular languages, it is not regular. Thus, the language $\Delta_{R_1}^{*}(S)$ is not regular. Hence, R_1 does not preserve regularity, if M has an immortal finite configuration. □

Observe that the strings in the set $\Delta_{R_1}^{*}(S)\cap 1^{+}\cdot 2^{+}\cdot \$\cdot \overline{\Sigma}^{*}\cdot Q\cdot \Sigma^{*}\cdot ¢$ are all irreducible mod R_1. Thus,

$$\mathrm{NF}_{R_1}(S)\cap 1^{+}\cdot 2^{+}\cdot \$\cdot \overline{\Sigma}^{*}\cdot Q\cdot \Sigma^{*}\cdot ¢ = \Delta_{R_1}^{*}(S)\cap 1^{+}\cdot 2^{+}\cdot \$\cdot \overline{\Sigma}^{*}\cdot Q\cdot \Sigma^{*}\cdot ¢,$$

and hence, R_1 does not even give regular sets of normal forms for regular languages, if M has an immortal finite configuration.

We would like to also prove the converse of this statement. However, for doing so we must consider all regular languages over Γ, not just the languages that only consist of strings which are encodings of configurations of M. To get around this problem, we introduce the finite string-rewriting system R_2, which constitutes the second part of the system R. It consists of the following 18 groups of monadic rules:

$$
\begin{array}{lll}
(1) & s1 & \to 0 \text{ for all } s \in \Gamma\backslash\{1\}; \\
(2) & s2 & \to 0 \text{ for all } s \in \Gamma\backslash\{1,2,\hat{c},\hat{d}\}; \\
(3) & s\hat{d} & \to 0 \text{ for all } s \in \Gamma\backslash\{1,2,\hat{d}\}; \\
(4) & 1\hat{d}\hat{d} & \to 0 \\
 & 2\hat{d}\hat{d}\hat{d} & \to 0 \\
(5) & s\hat{c} & \to 0 \text{ for all } s \in \Gamma\backslash\{1,2\}; \\
(6) & s\$ & \to 0 \text{ for all } s \in \Gamma\backslash\{2,\hat{c},\hat{d}\}; \\
(7) & s\bar{a} & \to 0 \text{ for all } \bar{a} \in \overline{\Sigma} \text{ and } s \in \Gamma\backslash(\overline{\Sigma} \cup \{\$,\bar{c},\bar{d}\}); \\
(8) & s\bar{q} & \to 0 \text{ for all } \bar{q} \in \overline{Q} \text{ and } s \in \Gamma\backslash(\overline{\Sigma} \cup \{\$,\bar{c},\bar{d}\}); \\
(9) & s\bar{d} & \to 0 \text{ for all } s \in \Gamma\backslash(\overline{\Sigma} \cup \{\$,\bar{d}\}); \\
(10) & s\bar{d}\bar{d}\bar{d} & \to 0 \text{ for all } s \in \overline{\Sigma} \cup \{\$\}; \\
(11) & s\bar{c} & \to 0 \text{ for all } s \in \Gamma\backslash(\overline{\Sigma} \cup \{\$\}); \\
(12) & sq & \to 0 \text{ for all } q \in Q \text{ and } s \in \Gamma\backslash(\overline{\Sigma} \cup \{\$,\bar{c},\bar{d}\}); \\
(13) & sa & \to 0 \text{ for all } a \in \Sigma \text{ and } s \in \Gamma\backslash(Q \cup \overline{Q} \cup \Sigma \cup \{d\}); \\
(14) & sd & \to 0 \text{ for all } s \in \Gamma\backslash(\overline{Q} \cup \Sigma \cup \{d\}); \\
(15) & sddd & \to 0 \text{ for all } s \in \overline{Q} \cup \Sigma; \\
(16) & s¢ & \to 0 \text{ for all } s \in \Gamma\backslash(Q \cup \overline{Q} \cup \Sigma \cup \{d\}); \\
(17) & sd_0 & \to 0 \text{ for all } s \in \Gamma\backslash\{¢,d_0\}; \\
(18) & \left.\begin{array}{ll} s0 & \to 0 \\ 0s & \to 0 \end{array}\right\} & \text{for all } s \in \Gamma.
\end{array}
$$

Obviously, R_2 is a finite and monadic string-rewriting system on Γ. Since each rule of R_2 has right-hand side 0, and since 0 acts as a zero because of the rules of (18), we can conclude that all the many critical pairs of R_2 resolve to 0. Thus, R_2 is also confluent. In addition, it has the following properties.

Lemma 6.4.
(a) $\forall s \in \Gamma\backslash\{d_0\} : d_0 s \to 0$.
(b) *The set* $\mathrm{IRR}(R_2)$ *consists of all the factors of the strings in the following language* $\mathrm{CONF} := 1^* \cdot (\{\hat{c},\hat{d}\} \cup (\{\lambda,\hat{c},\hat{d}\} \cdot (2 \cdot \{\lambda,\hat{c},\hat{d},\hat{d}\hat{d}\})^+)) \cdot \$ \cdot (\{\lambda,\bar{c},\bar{d},\bar{d}\bar{d}\} \cdot \overline{\Sigma})^* \cdot \{\lambda,\bar{c},\bar{d},\bar{d}\bar{d}\} \cdot ((\overline{Q} \cdot \{\lambda,d,dd\}) \cup Q) \cdot (\Sigma \cdot \{\lambda,d,dd\})^* \cdot ¢ \cdot d_0^* \cup \{0\}$.

Proof.
(a) This is easily seen from the rules of R_2.
(b) First of all, it can be checked easily that $\mathrm{CONF} \subseteq \mathrm{IRR}(R_2)$. On the other hand, each string $w \in \mathrm{IRR}(R_2)$ is a factor of a string from CONF. □

Observe that $1^+ \cdot 2^+ \cdot \$ \cdot \overline{\Sigma}^* \cdot Q \cdot \Sigma^* \cdot ¢ \subseteq \mathrm{CONF}$. Thus, all the strings in the language $\Delta^*_{R_1}(S) \cap 1^+ \cdot 2^+ \cdot \$ \cdot \overline{\Sigma}^* \cdot Q \cdot \Sigma^* \cdot ¢$ considered in the proof of Lemma 6.3 are irreducible mod R_2. Actually, if uqv is a configuration of the Turing machine M, then $\Delta^*_{R_1}(1^\ell 2^k \$ \bar{u} q v ¢ d_0^*) \subseteq \mathrm{CONF}$ for all $\ell, k \geqslant 1$. In fact, if $w \in \mathrm{IRR}(R_2)$, then $\Delta^*_{R_1}(w) \subseteq \mathrm{IRR}(R_2)$ as can be checked easily.

Finally, we consider the combined system $R := R_1 \cup R_2$.

Lemma 6.5. *R is a finite, length-reducing, and confluent system.*

Proof. It remains to verify that all overlaps between rules of R_1 and rules of R_2 resolve to 0. We look at each group of rules of R_2 in turn.
(1) If $(1\ell_1 \to r) \in R_1$, then $r = 1 \cdot r_1$ for some $r_1 \in \Gamma^*$, and hence,

$$\begin{tikzcd}
s1\ell_1 \arrow[r] \arrow[d] & s1r_1 \arrow[r] & 0 \cdot r_1 \arrow[dl, "*"] \\
0 \cdot \ell_1 \arrow[r, "*"] & 0 &
\end{tikzcd}$$

If $(\ell_1 s \to r) \in R_1$ for some $s \in \Gamma\backslash\{1\}$, then $r = r_1 t$ for some $t \in \Gamma\backslash\{1\}$.

Hence:

$$\begin{tikzcd}
\ell_1 s1 \arrow[r] \arrow[dd] & \ell_1 0 \arrow[dr, "*"] & \\
 & & 0 \\
r_1 t1 \arrow[r] & r_1 0 \arrow[ur, "*"] &
\end{tikzcd}$$

(2) If $(2\ell_1 \to r) \in R_1$, then $r = 2r_1$, and if $(\ell_1 s \to r) \in R_1$ for some $s \in \Gamma\backslash\{1,2,\hat{c},\hat{d}\}$, then $r = r_1 t$ for some $t \in \Gamma\backslash\{1,2,\hat{c},\hat{d}\}$. Hence, the resulting critical pairs resolve as in (1).

The remaining cases can be treated analogously. □

From Lemma 6.3 and the remark following Lemma 6.4 we see the following.

Corollary 6.6. *If the Turing machine M has an immortal finite configuration, then R does not preserve regularity. In fact, in this situation there exists a regular language $S \subseteq \Gamma^*$ such that not even the set $\mathrm{NF}_R(S)$ is regular.*

It remains to prove the converse of this corollary. Since the system R_2 can be used to reduce every string to 0 that does not represent a (piece of a) configuration of the Turing machine M, we can restrict our attention essentially to regular languages that only contain strings which are encodings of configurations of M. However, even this does not solve our problem, since even if the Turing machine M has no immortal finite configuration, it is still not clear how to prove that the set of irreducible descendants of an arbitrary regular set of configurations is itself regular. Therefore, we will prove the following weaker statement only: if the Turing machine M is strongly bounded, then $\mathrm{NF}_R(S)$ is a regular set for each regular language $S \subseteq \Gamma^*$.

First observe that it suffices to look at regular languages $S \subseteq \Gamma^*$ that are fairly restricted. Let $S \subseteq \Gamma^*$ be a regular language. Then $S = S_1 \cup S_2$, where $S_1 := S \cap \mathrm{IRR}(R_2)$ and $S_2 := S \cap \mathrm{RED}(R_2)$. Since R_2 is a finite string-rewriting system, we see that S_1

and S_2 are both regular sets. Now $\mathrm{NF}_R(S) = \mathrm{NF}_{R_1}(S_1) \cup \mathrm{NF}_R(S_2)$, and

$$\mathrm{NF}_R(S_2) = \begin{cases} \emptyset & \text{if } S_2 = \emptyset, \\ \{0\} & \text{if } S_2 \neq \emptyset. \end{cases}$$

Thus, $\mathrm{NF}_R(S)$ is regular if and only if $\mathrm{NF}_{R_1}(S_1)$ is regular. Hence, in the following we can restrict our attention to regular sets S that are contained in $\mathrm{IRR}(R_2)$. Thus, by Lemma 6.4(b) we only have to deal with regular sets of factors of the language CONF.

Let $S \subseteq \mathrm{IRR}(R_2)$ be a regular language. Again we partition S into two subsets $S_1 := S \cap \Gamma^* \cdot d_0^+$ and $S_2 := S \cap (\Gamma \backslash \{d_0\})^*$. Then $S = S_1 \cup S_2$, and $\mathrm{NF}_R(S) = \mathrm{NF}_{R_1}(S_1) \cup \mathrm{NF}_{R_1}(S_2)$. With S also S_1 and S_2 are regular languages.

Lemma 6.7. *If $S_2 \subseteq \mathrm{IRR}(R_2) \cap (\Gamma \backslash \{d_0\})^*$ is a regular language, then so is the language $\mathrm{NF}_{R_1}(S_2)$.*

Proof. Since a string $w \in S_2$ does not contain any occurrences of the symbol d_0, a reduction sequence $w = w_0 \to_{R_1} w_1 \to_{R_1} \cdots \to_{R_1} w_n \in \mathrm{IRR}(R)$ can contain at most a single application of a rule from group (1) of R_1.

A sequence $w = w_0 \to_{R_1} w_1 \to_{R_1} \cdots \to_{R_1} w_m$ of reduction steps is called *connected* if, for $i = 1, \ldots, m-1, w_{i-1} = u_{i-1}\ell_{i-1}v_{i-1}, w_i = u_{i-1}r_{i-1}v_{i-1} = u_i\ell_i v_i$, and $w_{i+1} = u_i r_i v_i$, where $(\ell_{i-1} \to r_{i-1}), (\ell_i \to r_i) \in R_1$, imply that r_{i-1} and ℓ_i have a nonempty overlap in w_i, that is, either $|u_i| < |u_{i-1}r_{i-1}| \leqslant |u_i\ell_i|$ or $|u_{i-1}| < |u_i\ell_i| < |u_{i-1}r_{i-1}|$. A connected sequence of R_1-reduction steps shifts $\hat{c}$ and $\bar{c}$ symbols to the right or it shifts $\hat{d}, \bar{d}$ and d symbols to the left. A sequence of this form can be followed by a single reduction step of a different form, for example, if $\delta(q_i, a_k) = (q_j, a_\ell)$, then

$$q_i a_k d a_r d a_{r_1} d \ldots d a_{r_s} d d ¢ \to_{R_1}^{s+1} q_i a_k d d a_r a_{r_1} \ldots a_{r_s} ¢ \to_{R_1} \bar{q}_j a_\ell a_r a_{r_1} \ldots a_{r_s} ¢.$$

However, since the strings considered do not contain any occurrences of the symbol d_0, those occurrences of the symbols $\hat{d}, \bar{d}$, and d that are used up during such a reduction sequence cannot be restored. Thus, a generalized sequential machine (gsm) G can be constructed that works as follows: while processing an input string $w \in \mathrm{IRR}(R_2)$ from left to right, G guesses an output string y and checks whether $w \to_{R_1}^* y$ holds. This can be done because of the observation above. A computation of G is accepting if and only if $w \in S_2, y \in \mathrm{IRR}(R_1)$, and $w \to_{R_1}^* y$. Hence, $G(S_2) = \mathrm{NF}_{R_1}(S_2)$, and so with S_2 also $\mathrm{NF}_{R_1}(S_2)$ is a regular language [16]. □

It remains to consider the regular language $S_1 = S \cap \Gamma^* \cdot d_0^+$. Let v denote the constant from the pumping lemma for regular languages that is associated with S_1. We partition S_1 even further as $S_1 = S_3 \cup S_4$, where

$$S_3 := S_1 \cap (\Gamma \backslash \{d_0\})^* \cdot \{d_0, d_0^2, \ldots, d_0^{v-1}\} \text{ and } S_4 := S_1 \cap \Gamma^* \cdot d_0^v.$$

Again S_3 and S_4 are regular languages, and

$$\mathrm{NF}_{R_1}(S_1) = \mathrm{NF}_{R_1}(S_3) \cup \mathrm{NF}_{R_1}(S_4).$$

Lemma 6.8. *If S_1 is a regular language, then so is the language* $\mathrm{NF}_{R_1}(S_3)$.

Proof. Obviously, $S_3 = \bigcup_{i=1}^{v-1} S_{3,i} \cdot d_0^i$ for some regular sets $S_{3,i} \subseteq (\Gamma \backslash \{d_0\})^*$, $i=1,\ldots,v-1$. Hence, $\mathrm{NF}_{R_1}(S_3) = \bigcup_{i=1}^{v-1} \mathrm{NF}_{R_1}(S_{3,i} \cdot d_0^i)$.

Let $i \in \{1,2,\ldots,v-1\}$. Then $\mathrm{NF}_{R_1}(S_{3,i} \cdot d_0^i) = \mathrm{NF}_{R_1}(\mathrm{NF}_{R_1}(S_{3,i}) \cdot d_0^i)$. By Lemma 6.7 $\mathrm{NF}_{R_1}(S_{3,i})$ is a regular language. Furthermore, it is easily seen that $\mathrm{NF}_{R_1}(S' \cdot d_0)$ is a regular language whenever S' is a regular language satisfying $S' \subseteq \mathrm{IRR}(R)$. It follows inductively that $\mathrm{NF}_{R_1}(S_{3,i} \cdot d_0^i)$ is a regular language for each $i = 1,\ldots,v-1$, and hence, $\mathrm{NF}_{R_1}(S_3) = \bigcup_{i=1}^{v-1} \mathrm{NF}_{R_1}(S_{3,i} \cdot d_0^i)$ is a regular language. □

Finally, we have to deal with the regular language $S_4 = S_1 \cap \Gamma^* \cdot d_0^v$. Since v is the constant that the pumping lemma yields for the language S_1, we see that, for each string $w \in (\Gamma \backslash \{d_0\})^*$ and each $m \geq v$, if $wd_0^m \in S_4$, then $wd_0^{m+\alpha \cdot r} \in S_4$ for some $\alpha \in \{1,\ldots,v\}$ and all integers $r \geqslant -1$.

Let S_5 denote the regular language

$$S_5 := \{w \in (\Gamma \backslash \{d_0\})^* \mid \exists m \geq v : wd_0^m \in S_4\}.$$

For $m \in \{0,1,\ldots,v-1\}$ and $\alpha \in \{1,\ldots,v\}$, define the languages $S_{5,m,\alpha}$ and $S_{4,m,\alpha}$ as follows:

$$S_{5,m,\alpha} := \{w \in S_5 \mid w \cdot d_0^m \cdot (d_0^\alpha)^* \subseteq S_4\}, \text{ and } S_{4,m,\alpha} := S_{5,m,\alpha} \cdot d_0^m \cdot (d_0^\alpha)^*.$$

From the considerations above we see that

$$S_4 = \bigcup_{m=0}^{v-1} \bigcup_{\alpha=1}^{v} S_{4,m,\alpha} \quad \text{and} \quad \mathrm{NF}_{R_1}(S_4) = \bigcup_{m=0}^{v-1} \bigcup_{\alpha=1}^{v} \mathrm{NF}_{R_1}(S_{4,m,\alpha}).$$

Obviously, each of the languages $S_{4,m,\alpha}$ is regular. Hence, it suffices to look at each of these languages separately. So let $m \in \{0,1,\ldots,v-1\}$ and $\alpha \in \{1,\ldots,v\}$.

Let $\Delta := Q \cup \Sigma \cup \overline{\Sigma} \cup \{\$, ¢\}$, and let $\varphi := \Gamma \to \Delta$ denote the morphism that is defined through

$$a \mapsto a \ (a \in \Delta)$$

$$\bar{q} \mapsto q \ (\bar{q} \in \overline{Q})$$

$$a \mapsto \lambda \ (a \in \{1,2,d,\bar{d},\hat{d},d_0,\hat{c},\bar{c},0\}).$$

We say that a string $w \in S_{5,m,\alpha}$ contains a configuration uqv of the Turing machine M, if $\varphi(w) = \$\bar{u}qv¢$.

Assume that the Turing machine M is strongly bounded. Then there exists an integer k such that, when starting in an arbitrary configuration uqv, M halts after at most k steps. Thus, only the suffix u_2 of u of length k and the prefix v_1 of v of length k are affected by the resulting computation. Hence, all possible computations of M can be described through the finite table

$$T = \{(uqv, u'q'v', i) \mid q, q' \in Q, u, v, u', v' \in \Sigma^*, |u|, |v|, |u'|, |v'| \leqslant k, \\ \text{and } i \leqslant k \text{ such that } uqv \vdash_M^i u'q'v'\}.$$

Let $w \in S_{5,m,\alpha}$. The case that w does not contain a configuration of M is easily dealt with. So let us assume that w contains a configuration $u_1u_2qv_1v_2$ of M, where $|u_2| = |v_1| = k$. If w does not begin with the symbol 1, or if $|w|_2 = 0$, then R_1 can simulate at most $\min\{|w|_{\hat{c}} + |w|_{\bar{c}}, k\}$ steps of M on wd_0^s, and in this case the numbers $|w|_1$ and $|w|_2$ are not changed at all (see the rules of the groups (3) to (5) of R_1). If, however, $|w|_1 = j_1$ and $|w|_2 = j_2$ for some $j_1, j_2 \geqslant 1$, then at most i additional occurrences of the symbols 1 and 2 can be created while reducing wd_0^s modulo R_1, where $i =$ number of simulated steps of $M - (|w|_{\hat{c}} + |w|_{\bar{c}})$, provided s is sufficiently large. Since M is strongly bounded by the constant k, we have $i \leqslant k$.

Hence, a gsm G can be constructed that works as follows: while processing an input string $x \in \mathrm{IRR}(R_2)$ from left to right, G guesses an output string y and checks the following three conditions:

1. Is $x = wd_0^s$ for some $w \in S_{5,m,\alpha}$ and some integer $s = m + \alpha \cdot r$?
2. Is $y = ud_0^t$ irreducible mod R_1, where $u \in (\Gamma \backslash \{d_0\})^*$ and $t \geqslant 0$?
3. Does $wd_0^\ell \to_{R_1}^* u$ hold for some $\ell \in \mathbb{N}$ satisfying the congruence $\ell + t \equiv m \bmod \alpha$, that is, $\ell + t = m + \alpha \cdot r$ for some $r \in \mathbb{N}$?

For the latter part of this test G uses the table T mentioned above for comparing the structure of the strings w and u. Obviously, G cannot compute the number ℓ of d_0-symbols that are used up in the reduction $wd_0^\ell \to_{R_1}^* u$, but it can determine whether the number $\ell + t$ satisfies the congruence $\ell + t = m + \alpha \cdot r$ for some $r \in \mathbb{N}$ by counting modulo α. For example, if G has already determined that ℓ_0 symbols d_0 are necessary to create the prefix v of u, and if $u = vau_1$ for some $a \in \Sigma$, then $2\ell_0$ symbols d_0 are needed to create the prefix va. However, if $\ell_0 \leqslant \alpha < 2\ell_0$, then $2\ell_0$ can be written as $\alpha + (2\ell_0 - \alpha)$, where $\ell_1 := 2\ell_0 - \alpha \leqslant \alpha$, and it suffices for G to remember the number ℓ_1. Thus, for all $w \in S_{5,m,\alpha}$ and $r \in \mathbb{N}$,

$$G(wd_0^{m+\alpha\cdot r}) = \Delta_{R_1}^*(w \cdot d_0^m \cdot (d_0^\alpha)^*) \cap \mathrm{IRR}(R_1).$$

Hence, it follows that

$$G(S_{4,m,\alpha}) = \Delta_{R_1}^*(S_{4,m,\alpha}) \cap \mathrm{IRR}(R_1) = \mathrm{NF}_{R_1}(S_{4,m,\alpha}),$$

and thus, $\mathrm{NF}_{R_1}(S_{4,m,\alpha})$ is regular, too. This completes the proof of the following lemma.

Lemma 6.9. *If the Turing machine M is strongly bounded, then* $\mathrm{NF}_R(S)$ *is a regular language for each regular language* $S \subseteq \Gamma^*$.

Given a single-tape Turing machine M that simulates a Minsky machine $\hat{M}$ as described in [15], we see from the discussion in Section 5 that M has an immortal finite configuration if and only if M is not strongly bounded. By Corollary 6.6 and

Lemma 6.9, $\mathrm{NF}_R(S)$ is regular for each regular language $S \subseteq \Gamma^*$ if and only if M is strongly bounded. Hence, Proposition 5.1 yields the following undecidability result.

Theorem 6.10. *The following problem is undecidable in general:*
Instance: A finite, length-reducing, and confluent string-rewriting system R on Γ.
Question: Is $\mathrm{NF}_R(S)$ a regular language for each regular language $S \subseteq \Gamma^*$?

This generalizes Theorem 9 of [12] to signatures containing unary function symbols only and possibly a single constant.

7. Conclusion

We have seen that for string-rewriting systems the property of preserving regularity is independent of the alphabet actually considered, i.e., by adding some free symbols to an alphabet considered, the property of a string-rewriting system to preserve regularity is not affected. Further, we have seen that for finite string-rewriting systems in general, the property of preserving regularity is undecidable in general. From these two results we derived the fact that the property of being regularity preserving is undecidable for term-rewriting systems, thus answering a question of Gyenizse and Vágvölgyi [14].

For finite, length-reducing, and confluent string-rewriting systems, we have shown that it is undecidable in general whether the set of descendants or the set of normal forms of a given regular language is again regular. Also we have seen that for a string-rewriting system of this form it is undecidable in general whether or not each regular language has a regular set of normal forms.

However, it remains open whether the property of being regularity preserving is also undecidable for the class of all finite string-rewriting systems that are length-reducing and confluent. In fact, we do not even know whether this is true for the class of all finite convergent string-rewriting systems, but we would certainly expect that. Also it remains the question of whether finite, length-reducing, and confluent systems presenting groups preserve regularity.

However, in the latter case we do at least know the following. If R is a finite, length-reducing, and confluent string-rewriting system on Σ such that the monoid M_R presented by $(\Sigma; R)$ is actually a group, then there exists a deterministic pushdown automaton P that, given a string $w \in \Sigma^*$ as input, computes the irreducible descendant w_0 of w mod R [20]. In fact, P can be realized in such a way that after processing the input w completely, it halts with w_0 in its pushdown store. For $L \subseteq \Sigma^*$, let $\mathrm{SC}_P(L)$ denote the language of final stack contents that P can generate given a string from L as input. Then $\mathrm{SC}_P(L) = \mathrm{NF}_R(L)$. If L is a regular language, then by a result of Greibach [13] also $\mathrm{SC}_P(L)$ is a regular language. Thus, in this situation the set of normal forms of a regular language is itself always regular.

Instead of asking whether the set of descendants or the set of normal forms of a regular language is again a regular language, we could also ask whether it is a context-free language. The results of this paper can easily be carried over to the corresponding variants of Problems 1–4. Indeed these questions could also be asked for other classes of languages.

Acknowledgements

The author wants to thank Klaus Madlener and Paliath Narendran for very fruitful discussions regarding the result presented in Section 6 and for pointing out the papers [15, 17].

References

[1] A.V. Anissimov, Group languages, Cybernetics 7 (1971) 594–601.
[2] R.V. Book, The power of the Church–Rosser property in string-rewriting systems, in: D.W. Loveland (Ed.), Proc. 6th Conf. on Automated Deduction, Lecture Notes in Computer Science, vol. 138, Springer, Berlin, 1982, pp. 360–368.
[3] R.V. Book, M. Jantzen, C. Wrathall, Monadic Thue systems, Theoret. Comput. Sci. 19 (1982) 231–251.
[4] R.V. Book, F. Otto, String-Rewriting Systems, Springer, New York, 1993.
[5] W.J. Brainerd, Tree generating regular systems, Inform. and Control 14 (1969) 217–231.
[6] J.-L. Coquidé, M. Dauchet, R. Gilleron, S. Vágvölgyi, Bottom-up tree pushdown automata: classification and connection with rewrite systems, Theoret. Comput. Sci. 127 (1994) 69–98.
[7] M. Dauchet, Simulation of Turing machines by a left-linear rewrite rule, in: N. Dershowitz (Ed.), Proc. Rewriting Techniques and Applications, Lecture Notes in Computer Science, vol. 355, Springer, Berlin, 1989, pp. 109–120.
[8] J. Gallier, R.V. Book, Reductions in tree replacement systems, Theoret. Comput. Sci. 37 (1985) 123–150.
[9] F. Gécseg, M. Steinby, Tree Automata, Akadémiai Kiadó, Budapest, 1984.
[10] F. Gécseg, M. Steinby, Tree languages, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, vol. III, Springer, Berlin, 1997, pp. 1–68.
[11] R. Gilleron, Decision problems for term rewriting systems and recognizable tree languages, in: C. Choffrut, M. Jantzen (Eds.), Proc. STACS'91, Lecture Notes in Computer Science, vol. 480, Springer, Berlin, 1991, pp. 148–159.
[12] R. Gilleron, S. Tison, Regular tree languages and rewrite systems, Fundam. Inform. 24 (1995) 157–175.
[13] S. Greibach, A note on pushdown store automata and regular systems, Proc. Amer. Math. Soc. 18 (1967) 263–268.
[14] P. Gyenizse, S. Vágvölgyi, Linear generalized semi-monadic rewrite systems effectively preserve recognizability, Theoret. Comput. Sci. 194 (1998) 87–122.
[15] P.K. Hooper, The undecidability of the Turing machine immortality problem, J. Symbol. Logic 31 (1966) 219–234.
[16] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, Reading, MA, 1979.
[17] A.J. Kfoury, J. Tiuryn, P. Urzyczyn, The undecidability of the semi-unification problem, Inform. and Comput. 102 (1993) 83–101.
[18] T. Kretschmer, A closure property of regular languages, Theoret. Comput. Sci. 61 (1988) 283–287.
[19] R.C. Lyndon, P.E. Schupp, Combinatorial Group Theory, Springer, Berlin, 1977.
[20] K. Madlener, F. Otto, Groups presented by certain classes of finite length-reducing string-rewriting systems, in: P. Lescanne (Ed.), Proc. Rewriting Techniques and Applications, Lecture Notes in Computer Science, vol. 256, Springer, Berlin, 1987, pp. 133–144.
[21] C. Ó'Dúnlaing, Finite and infinite regular Thue systems, Ph.D. Thesis, University of California, Santa Barbara, 1981.

[22] F. Otto, Some undecidability results for non-monadic Church–Rosser Thue systems, Theoret. Comput. Sci. 33 (1984) 261–278.

[23] F. Otto, When is an extension of a specification consistent? Decidable and undecidable cases, J. Symbol. Comput. 12 (1991) 255–273.

[24] J. Sakarovitch, Syntaxe des langages de Chomsky, essai sur le déterminisme, Thèse de doctorat d'état de l'université Paris VII, 1979.

[25] K. Salomaa, Deterministic tree pushdown automata and monadic tree rewriting systems, J. Comput. Syst. Sci. 37 (1988) 367–394.

[26] G. Sénizergues, Some decision problems about controlled rewriting systems, Theoret. Comput. Sci. 71 (1990) 281–346.

[27] M. Wirsing, Algebraic specification, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, vol. B.: Formal Models and Semantics, Elsevier, Amsterdam, 1990, pp. 675–788.