Codes modulo finite monadic string-rewriting systems

Friedrich Otto
Fachber. Mathematik/Informatik, Gesamthochschule Kassel, Postfach 101 380, 34109 Kassel, Germany

Paliath Narendran
Department of Computer Science, Institute for Programming and Logics, State University of New York, Albany, NY 12222, USA

Abstract

A set $C \subseteq \Sigma^*$ is called a code modulo a string-rewriting system $T$ if, for all $u_1, u_2, \ldots, u_k, w_1, w_2, \ldots, w_m \in C$, $v_1 v_2 \cdots v_k \rightarrow^* T w_1 w_2 \cdots w_m$ implies that $k = m$ and $v_i = w_i$, $i = 1, \ldots, k$. Here we show that it is decidable whether a regular set is a code modulo $T$, when $T$ is a finite string-rewriting system that is monadic and confluent, or that is special and $\lambda$-confluent.

1. Introduction

A set of strings $U = \{u_i \mid i \in I\} \subseteq \Sigma^*$ is called independent if, for each $i \in I$, $u_i \notin (U - \{u_i\})^*$, i.e., the string $u_i$ cannot be written as a product of strings from $U - \{u_i\}$. The set $U$ is a code if, for all $k, m \geq 0$ and all $v_1, \ldots, v_k, w_1, \ldots, w_m \in U$, $v_1 v_2 \cdots v_k = w_1 w_2 \cdots w_m$ implies that $k = m$ and $v_i = w_i$, $i = 1, 2, \ldots, k$, i.e., each string from $U^*$ has a unique factorization as a product of strings from $U$. Obviously, each code is an independent set, but not necessarily vice versa. For example, the set $A := \{ab, ba, aba\}$ is independent, but it is not a code, since $(ab)(aba) = (aba)(ba)$.

Here we are interested in the following generalization of these notions. Let $T$ be a string-rewriting system on $\Sigma$, and let $\leftrightarrow_T^*$ denote the Thue congruence induced by $T$. A set of strings $U = \{u_i \mid i \in I\} \subseteq \Sigma^*$ is called independent mod $T$ if, for each $i \in I$, $u_i$ is not congruent to any string from $(U - \{u_i\})^*$, and it is called a code mod $T$ if, for all $k, m \geq 0$...
and all \( v_1, v_2, \ldots, v_k, w_1, w_2, \ldots, w_m \in U \), \( v_1 v_2 \ldots v_k \leftrightarrow^* w_1 w_2 \ldots w_m \) implies that \( k = m \) and \( v_i = w_i, \ i = 1, 2, \ldots, k \). Algebraically these notions can be interpreted as follows. Let \( M_T \) denote the factor monoid \( \Sigma^*/\leftrightarrow^* \). Then the set \( U \) is independent mod \( T \) if and only if, for each \( i \in I, u_i \) does not belong to the submonoid of \( M_T \) that is generated by \( U - \{ u_i \} \), and \( U \) is a code mod \( T \) if and only if the submonoid of \( M_T \) that is generated by \( U \) is in fact freely generated by \( U \).

It is decidable whether a regular set \( U \) is independent or whether it is a code \([3, 5]\). On the other hand, there exists a finite string-rewriting system \( T \) that is even length-reducing and confluent such that it is undecidable in general whether or not a finite set \( U \) is independent mod \( T \) \([11, \text{Theorem 3.4}]\). Further, given a finite, length-reducing and confluent string-rewriting system \( T \) on some alphabet \( \Sigma_2 \) and a subalphabet \( \Sigma_1 \) of \( \Sigma_2 \), it is undecidable in general whether or not \( \Sigma_1 \) is a code mod \( T \) \([12, \text{Theorem 3.4}]\). The latter undecidability result also holds for the class of finite monadic string-rewriting systems that are weakly confluent \([13]\). On the other hand, the property that a finite set \( U = \{ u_1, \ldots, u_n \} \) is independent mod \( T \) can be expressed by a linear sentence \([2]\). Thus, if \( T \) is a finite string-rewriting system that is (i) monadic and confluent \([2]\), that is (ii) monadic and \( \lambda \)-confluent, and that presents a group \([10]\), or that is (iii) special and \( \lambda \)-confluent \([14]\), then it is decidable in polynomial time whether or not a finite set is independent mod \( T \).

Here we show that it is decidable whether a regular set \( U \) is a code modulo a finite string-rewriting system \( T \) provided \( T \) is (i) monadic and confluent or (ii) special and \( \lambda \)-confluent. Actually, we show that the following technical problem is decidable:

**Instance:** A finite string-rewriting system \( T \) on some alphabet \( \Sigma \) such that (i) \( T \) is monadic and confluent or (ii) \( T \) is special and \( \lambda \)-confluent, and a regular set \( R \subseteq \Sigma^* \) (specified in a suitable way).

**Question:** Do there exist strings \( u, v \in R \) such that \( u \neq v \), but \( u \leftrightarrow^*_T v \)?

The decidability of this problem implies the decidability of the former as follows. Let \( U \subseteq \Sigma^* \) be a regular set. If \( U \) is not a code (and this is decidable), then certainly \( U \) is not a code mod \( T \). If, however, \( U \) is a code, then \( U \) is not a code mod \( T \) if and only if there are strings \( u, v \in U^* \) such that \( u \neq v \) and \( u \leftrightarrow^*_T v \). Since \( U^* \) is regular, this is decidable by the latter result.

In Section 2 we first restate some of the fundamental definitions and notions regarding string-rewriting systems in short to establish notation. For a thorough introduction to string-rewriting systems the interested reader is asked to consult the literature, e.g., the recent monograph \([4]\). Then we state our main result formally. Our proof, presented in Section 3, effectively reduces the problem considered to some decidable problems on regular and deterministic context-free languages. For this part we assume the reader to be familiar with the basic theory of finite-state acceptors and pushdown automata as, e.g., presented in \([6]\).
2. Special and monadic string-rewriting systems

Let $\Sigma$ be a finite alphabet. Then $\Sigma^*$ denotes the set of strings over $\Sigma$ including the empty string $\lambda$. For $w \in \Sigma^*$, $|w|$ denotes the length of $w$. A string-rewriting system $T$ on $\Sigma$ is a subset of $\Sigma^* \times \Sigma^*$, the elements of which are called (rewrite) rules. By $\text{dom}(T)$ we denote the set $\{\ell \mid \exists r : (\ell \rightarrow r) \in T\}$ of all left-hand sides of rules, and by $\text{range}(T)$ we denote the set $\{r \mid \exists \ell : (\ell \rightarrow r) \in T\}$ of all right-hand sides. The system $T$ is called length-reducing if $|\ell| > |r|$ holds for each rule $(\ell \rightarrow r) \in T$; it is called monadic if it is length reducing and $\text{range}(T) \subseteq \Sigma \cup \{\lambda\}$, and it is called special if it is length-reducing and $\text{range}(T) = \{\lambda\}$.

The single-step reduction relation induced by $T$ is denoted by $\rightarrow_T$. Its reflexive transitive closure $\rightarrow^*_T$ is the reduction relation induced by $T$, and its reflexive, symmetric and transitive closure $\leftrightarrow^*_T$ is the Thue congruence generated by $T$.

A string $w \in \Sigma^*$ is reducible (mod $T$) if there exists a string $z \in \Sigma^*$ such that $w \rightarrow_T z$; otherwise, $w$ is irreducible (mod $T$). The set of irreducible strings, which is denoted by $\text{IRR}(T)$, is a regular set, for which a deterministic finite-state acceptor can be constructed in polynomial time from $T$ whenever the system $T$ is finite.

For $w \in \Sigma^*$, $D_T^*(w) = \{u \in \Sigma^* \mid w \rightarrow^*_T u\}$ is the set of descendants of $w$, and for $L \subseteq \Sigma^*$, $A_T^*(L) = \bigcup_{w \in L} D_T^*(w)$. If $T$ is monadic, and $L \subseteq \Sigma^*$ is a regular set, then the set $A_T^*(L)$ is regular, too [8]. If, in addition, $T$ is finite, then a nondeterministic finite-state acceptor for the set $A_T^*(L)$ can be constructed in polynomial time from $T$ and from a finite-state acceptor for the set $L$ ([2], for a detailed presentation of this construction see [4]).

A string-rewriting system $T$ is noetherian if there does not exist an infinite sequence of reductions of the form $w_0 \rightarrow_T w_1 \rightarrow_T w_2 \rightarrow_T \ldots$, it is confluent if, for all $u, v \in \Sigma^*$, $u \rightarrow^*_T v$ implies that $D_T^*(u) \cap D_T^*(v) \neq \emptyset$, i.e., $u$ and $v$ have a common descendant. If $T$ is noetherian and confluent, each congruence class $[w]_T := \{z \in \Sigma^* \mid z \leftrightarrow^*_T w\}$ contains a unique irreducible string. Obviously, if $T$ is length-reducing, then it is noetherian; in fact, an irreducible descendant of $w$ can then be computed in time bounded above by a polynomial in $|w|$ and $\text{size}(T)$ ($:= \Sigma_{(\ell \rightarrow r) \in T} (|\ell| + |r|)$) [1].

Sometimes a string-rewriting system $T$ considered is not confluent in general, but only on certain congruence classes. Here $T$ is said to be confluent on $[w]_T$ if $[w]_T$ contains a single irreducible string only. We say that $T$ is $\lambda$-confluent if it is confluent on $[\lambda]_T$, and a monadic system $T$ is called weakly confluent if it is confluent on $[\lambda]_T$ for all $\lambda \in \Sigma \cup \{\lambda\}$.

Finally, a string-rewriting system $T$ is called normalized if, for each rule $(\ell \rightarrow r) \in T$, $\ell \in \text{IRR}(T - \{\ell \rightarrow r\})$ and $r \in \text{IRR}(T)$. Given a finite string-rewriting system $T$ that is monadic and (weakly) confluent, or that is special and $\lambda$-confluent, an equivalent system $T_0$ of the same form can be constructed in polynomial time such that $T_0$ is normalized [7, 9, 14]. Thus, we can restrict our attention to finite normalized systems in these cases.
Concerning finite monadic and weakly confluent string-rewriting systems we just want to restate the following undecidability result from [13].

**Theorem 2.1.** The following problem is undecidable in general.

*Instance:* A finite, monadic, and weakly confluent string-rewriting system \( T \) on some alphabet \( \Sigma \), and a subalphabet \( \Gamma \subseteq \Sigma \).

*Question:* Is \( \Gamma \) a code mod \( T \)?

In the following we will only be concerned with finite string-rewriting systems that are either monadic and confluent or special and \( \lambda \)-confluent. We want to establish the following result.

**Theorem 2.2.** The following problem is decidable.

*Instance:* A finite string-rewriting system \( T \) that is monadic and confluent or that is special and \( \lambda \)-confluent, and a regular set \( U \subseteq \Sigma^* \).

*Question:* Is \( U \) a code mod \( T \)?

Comparing Theorem 2.2 to Theorem 2.1 we see the following. For a finite monadic string-rewriting system \( T \) the property of weak confluence is not even powerful enough to enforce that it becomes decidable whether a finite subset of the given alphabet \( \Sigma \) is a code mod \( T \), while the stronger property of confluence is sufficient to solve the more general problem of deciding whether or not an arbitrary regular set \( U \subseteq \Sigma^* \) is a code. Our goal is to prove Theorem 2.2.

While each congruence class of a monadic and confluent string-rewriting system contains a unique irreducible string, this is not true in general for special and \( \lambda \)-confluent systems. However, for these systems we do at least have the following normal form theorem.

**Proposition 2.3** (Otto and Zhang [14]). Let \( T \) be a special string-rewriting system on \( \Sigma \).

(a) For each string \( u \in \Sigma^+ \), there is a unique factorization of the form

\[
    u = u_0 a_1 u_1 \ldots a_m u_m
\]

for some \( m \geq 0 \), where

(i) \( u_0, u_1, \ldots, u_m \) are maximal invertible factors of \( u \) (some of which may be empty), and

(ii) \( a_1, \ldots, a_m \in \Sigma \).

(b) Let \( u, v \in \Sigma^+ \), and let \( u = u_0 a_1 u_1 \ldots a_m u_m \) and \( v = v_0 b_1 v_1 \ldots b_n v_n \) be the factorizations of \( u \) and \( v \), respectively. Then \( u \mathrel{\xrightarrow{+}} v \) if and only if

(i) \( n = m \),

(ii) \( a_i = b_i \) for \( i = 1, \ldots, m \), and

(iii) \( u_i \mathrel{\xrightarrow{+}} v_i \) for \( i = 0, 1, \ldots, m \).
Accordingly, the above factorization of a string \( u \) is called the \textit{normal form} of \( u \). Here a factor \( x \) of a string \( u \) is a \textit{maximal invertible factor} of \( u \) if \( x \) is invertible mod \( T \), i.e., \( xy \leftrightarrow^* \lambda \leftrightarrow^* yx \) for some \( y \in \Sigma^* \), and no invertible factor \( z \) of \( u \) properly contains \( x \).

Actually, if \( T \) is a special system that is finite and \( \lambda \)-confluent, then the normal form of a string \( u \) can be determined in polynomial time. In fact, let \( U(T) \) denote the set of invertible \( \Sigma^* \) mod \( T \), i.e., \( w \in U(T) \) if and only if \( zw \leftrightarrow^* \lambda \leftrightarrow^* wz \) for some \( z \in \Sigma^* \), and let \( D_T \) be the following subset of \( U(T) \):

\[
D_T := \{ u \in \Sigma^* \mid u \cdot \Sigma^* \cap \text{dom}(T) \neq \emptyset \neq \Sigma^* \cdot u \cap \text{dom}(T), \text{ and no proper nonempty prefix } v \text{ of } u \text{ satisfies } \Sigma^* \cdot v \cap \text{dom}(T) \neq \emptyset \}.
\]

Then \( D_T \) is a finite biprefix code that can easily be obtained from \( T \). Further, \( U(T) \cap \text{IRR}(T) \subseteq D_T^\perp \), i.e., an irreducible string \( u \) is invertible mod \( T \) if and only if \( u \in D_T^\perp \). Thus, it is easily decidable whether or not an irreducible string is invertible mod \( T \). Further, for each \( u \in D_T \), we can obtain a string \( v \in \Sigma^* \), \( |v| < \mu \), such that \( uv \leftrightarrow^* \lambda \leftrightarrow^* vu \), i.e., we have a mapping \( ^{-1} : U(T) \cap \text{IRR}(T) \rightarrow U(T) \) such that, for all \( w \in U(T) \cap \text{IRR}(T) \), \( \lambda w^{-1} \leftrightarrow^* \Lambda w^{-1} w \), and \( |w^{-1}| < \mu \cdot |w| \). Here \( \mu = \max \{|f| \mid f \in \text{dom}(T)\} \). Finally, we can construct a generalized sequential machine (gsm) \( GNF \) that, given an irreducible string \( u \in \Sigma^* \) as input such that \( u_0 a_1 u_1 \ldots a_m u_m \) is the normal form of \( u \), computes the string \( u_0 b_1 u_1 \ldots b_m u_m \), where \( \Gamma := \{ b_i \mid a_i \in \Sigma \} \) is a new alphabet in one-to-one correspondence to \( \Sigma \).

These technical results, which are taken from [14], will be useful tools for proving Theorem 2.2 in the case of finite, special, and \( \lambda \)-confluent string-rewriting systems.

3. The proof

As shown in the introduction it suffices to establish the following result in order to prove Theorem 2.2.

\textbf{Theorem 3.1.} The following problem is decidable:

\textbf{Instance:} A finite string-rewriting system \( T \) that is monadic and confluent or that is special and \( \lambda \)-confluent, and a regular set \( R \subseteq \Sigma^* \).

\textbf{Question:} \( \exists u, v \in R : u \neq v \), but \( u \leftrightarrow^* \lambda \leftrightarrow^* v \)?

Our proof will be based on the following two observations. Let \( R \subseteq \Sigma^* \). The \textit{syntactic congruence} \( \text{syn}(R) \) of \( R \) is defined by

\[
\text{syn}(R) = \{ (u, v) \mid \forall x, y \in \Sigma^* : xuy \in R \text{ iff } xyv \in R \},
\]

and the factor monoid \( \Sigma^*/\text{syn}(R) \) is known as the \textit{syntactic monoid} of \( R \). It is well known that this monoid is finite if and only if the set \( R \) is regular. In this case, let
denote the cardinality of the syntactic monoid of \( R \), and let \( \text{SUBS}(R) \) be the following set:

\[
\text{SUBS}(R) := \{ v \in \Sigma^* \mid \exists u, w \in \Sigma^*: uvw \in R \},
\]

i.e., \( \text{SUBS}(R) \) is the set of all factors of strings of \( R \). The following lemma now expresses our first observation.

**Lemma 3.2.** Let \( T \) be a string-rewriting system on \( \Sigma \), and let \( R \subseteq \Sigma^* \) be a regular set. If there exist \( \sigma_R + 1 \) distinct strings \( w_1, w_2, \ldots, w_{\sigma_R + 1} \in \text{SUBS}(R) \) such that \( w_i \leftrightarrow_T^* w_j \) for all \( i, j, 1 \leq i < j \leq \sigma_R + 1 \), then there are two distinct strings \( x, y \in R \) such that \( x \leftrightarrow_T^* y \).

**Proof.** Since \( \sigma_R + 1 > |\Sigma^*/\text{syn}(R)| \), there are indices \( i, j, 1 \leq i < j \leq \sigma_R + 1 \), such that \( (w_i, w_j) \in \text{syn}(R) \). Since \( w_i \in \text{SUBS}(R) \), \( u_w, v \in R \) for some \( u, v \in \Sigma^* \). Since \( (w_i, w_j) \in \text{syn}(R) \), this implies that \( u_w v \in R \), too. Further, from \( w_i \leftrightarrow_T^* w_j \) we obtain \( u_w v \leftrightarrow_T^* u_w v \). However, since \( w_i \neq w_j \), also \( u_w v \neq u_w v \), i.e., \( u_w v \) and \( u_w v \) are two distinct strings from \( R \) that are congruent mod \( T \).

Secondly, the statement

\[(1) \exists u, v \in R: u \neq v \text{ and } u \leftrightarrow_T^* v\]

is equivalent to the disjunction of the following two statements, since if \( u \) and \( v \) exist with the above properties, then we can compare them using the lexicographical ordering on \( \Sigma^* \):

\[(2) \exists u, v \in \Sigma^* \exists a \in \Sigma: u \in R, uav \in R, \text{ and } u \leftrightarrow_T^* uav, \text{ or} \]

\[(3) \exists u, v, w \in \Sigma^* \exists a, b \in \Sigma, a \neq b: uav \in R, ubw \in R, \text{ and } uav \leftrightarrow_T^* ubw.\]

We shall now deal with the latter two statements separately. So let \( T \) be a finite string-rewriting system on \( \Sigma \) that is either monadic and confluent or that is special and \( \lambda \)-confluent, and let \( R \subseteq \Sigma^* \) be a regular set. For the following considerations we fix the system \( T \) and the set \( R \), and take \( n := |\Sigma| \). For \( a \in \Sigma \), we define a language \( I_a \) as follows:

\[
I_a := \{ u_0 \#_a v_0 \mid u_0, v_0 \in \text{IRR}(T), \text{ and } \exists u \in R \exists v \in \Sigma^*: u \leftrightarrow_T^* u_0, v \leftrightarrow_T^* v_0, \text{ and } uav \in R \},
\]

where \( \#_a \) is an additional symbol.

**Lemma 3.3.** The language \( I_a \) is regular, and from \( R, T \) and \( a \in \Sigma \), a finite-state acceptor for \( I_a \) can be constructed effectively.

**Proof.** Let \( H_a := \{ u\#_a v \mid u \in R, v \in \Sigma^* \text{ and } uav \in R \} \). Then \( H_a \) is a regular set, for which a finite-state acceptor can easily be constructed. Since the string-rewriting system \( T \) is
monadic (or even special), the set \( \Delta^*_{T}(H_a) \) is regular, and a finite-state acceptor can be constructed that accepts this set. Hence, the set

\[
\Delta^*_{T}(H_a) \cap IRR(T) \cdot \{\#_{a}\} \cdot IRR(T) = I_a
\]

is also regular, and a finite-state acceptor for this set can be constructed effectively.  

Let \( J_a \) denote the set

\[
J_a := \{ u_0 a v_0 | u_0, v_0 \in IRR(T), \text{ and } \exists u \in R \exists v \in \Sigma^* : u \rightarrow^* u_0, v \rightarrow^* v_0, \text{ and } uav \in R \}.
\]

Then \( J_a = \Psi_a(I_a) \), where \( \Psi_a : (\Sigma \cup \{\#_{a}\})^* \rightarrow \Sigma^* \) is the morphism induced by \( b \mapsto b (b \in \Sigma) \) and \( \#_{a} \mapsto a \). Thus, we conclude the following.

**Corollary 3.4.** The language \( J_a \) is regular, and a finite-state acceptor for \( J_a \) can be constructed effectively.

Let \( u_0 a v_0 \in J_a \). Then \( u_0 \) and \( v_0 \) are irreducible mod \( T \). If \( u_0 a v_0 \) admits a reduction sequence of length \( m \), i.e., there is a reduction sequence of the form

\[
u_0 a v_0 = z_0 \rightarrow^T z_1 \rightarrow^T \cdots \rightarrow^T z_m,
\]

then \( u_0 \) and \( v_0 \) can be factored as \( u_0 = x_m \cdots x_1 x_0 \) and \( v_0 = y_0 y_1 \cdots y_m \) such that, for all \( i = 1, \ldots, m \),

\[
z_{i-1} = x_m \cdots x_i x_{i-1} a_{i-1} y_{i-1} y_i \cdots y_m \rightarrow^T x_m \cdots x_i a_i y_i \cdots y_m = z_i
\]

for some \( a_1, \ldots, a_m \in \Sigma \cup \{\lambda\} \), where \( a_0 = a \).

**Lemma 3.5.** If the language \( J_a \) contains a string that admits a reduction sequence of length \( (n+1) \cdot \sigma_R + 1 \), then there are two distinct strings \( x, y \in R \) with \( x \rightarrow^T y \).

**Proof.** Assume that the string \( u_0 a v_0 \in J_a \) admits a reduction sequence of length \( m := (n+1) \cdot \sigma_R + 1 \). Then \( u_0 = x_m \cdots x_1 x_0 \) and \( v_0 = y_0 y_1 \cdots y_m \) such that

\[
u_0 a v_0 = x_m \cdots x_1 x_0 a y_0 y_1 \cdots y_m \rightarrow^T x_m \cdots x_1 a y_1 \cdots y_m \rightarrow^T \cdots \rightarrow^T x_m a_m y_m
\]

for some \( a_1, \ldots, a_m \in \Sigma \cup \{\lambda\} \). Since \( u_0 a v_0 \in J_a \), there are strings \( u \in R \) and \( v \in \Sigma^* \) such that \( u \rightarrow^* u_0 \), \( v \rightarrow^* v_0 \) and \( uav \in R \). \( T \) being monadic implies that \( u \) and \( v \) can be factored as \( u = f_m \cdots f_1 f_0 \) and \( v = g_0 g_1 \cdots g_m \) such that \( f_i \rightarrow^T x_i \) and \( g_i \rightarrow^T y_i \), \( i = 0, 1, \ldots, m \). Thus,

\[
uav = f_m \cdots f_1 f_0 a g_0 g_1 \cdots g_m,
\]

and

\[
f_{i-1} \cdots f_0 a g_0 \cdots g_{i-1} \rightarrow^* x_{i-1} \cdots x_0 a y_0 \cdots y_{i-1} \rightarrow^* a_i
\]

where \( x_0 = a_0 \), \( y_0 = a_0 \), \( a_1, \ldots, a_m \in \Sigma \cup \{\lambda\} \), and \( a_0 \neq a \).
for all $i = 1, 2, \ldots, m$. Since $m > (n+1) \cdot \sigma_R$, there are indices $0 < i_1 < i_2 < \cdots < i_{\sigma_R} < i_{\sigma_R+1} \leq m$ such that $a_{i_1} = a_{i_2} = \cdots = a_{i_{\sigma_R+1}}$. Let

$$A := \{ f_{j_1} \cdots f_{j_{\sigma_R}} a_{g_1} \cdots g_{j_{\sigma_R+1}} | j_1, j_2, \ldots, j_{\sigma_R+1} \}.$$ 

Then $A \subseteq \text{SUBS} \{ \{ u_{av} \} \} \subseteq \text{SUBS}(R)$. All strings from $A$ are congruent to $a_{i_1}$, and, for all $k = 1, 2, \ldots, m$, $f_k g_k \neq \lambda$ implying that $|A| = \sigma_R + 1$. Thus, Lemma 3.2 applies. \hfill \Box

If the string $u_0 a v_0 \in J_a$ ($u_0, v_0 \in \text{IRR}(T)$) does not admit a reduction sequence of length $m$ for some fixed integer $m$, then $u_0$ and $v_0$ can be factored as $u_0 = u_1 u_2$ and $v_0 = v_2 v_1$, where $|u_1|, |v_2| \leq (m-1) \cdot \mu$, $u_2 a v_2 \rightarrow^* w$ and $u_1 w v_1 \in \text{IRR}(T)$. The reason is the fact that in a reduction sequence of length at most $m-1$ only a suffix $u_2$ of $u_0$ and a prefix $v_2$ of $v_0$ can be involved that are of length not exceeding $(m-1) \cdot \mu$, since $u_0$ and $v_0$ are irreducible.

For $a \in \Sigma$, we define the language $L_a$ as follows:

$$L_a := \{ u_0 a v_0 | u_0 \# a v_0 \in I_a \text{ and } u_0 a v_0 \rightarrow^*_T u_0 \}. $$

Observe that if $u_0 a v_0 \in L_a$, then there exist strings $u \in R$ and $v \in \Sigma^*$ such that $u \rightarrow^*_R u_0$, $v \rightarrow^*_T v_0$, $u a v \in \text{R}$, and $u_0 a v_0 \rightarrow^*_T u_0$, i.e., $u a v \rightarrow^*_T u_0 a v_0 \rightarrow^*_T u_0 \rightarrow^*_T u$ and $u a v \in R$. Thus, if $L_a \neq \emptyset$ for some $a \in \Sigma$, then statement (2) holds for $T$ and $R$.

**Lemma 3.6.** Let $m \in \mathbb{N}$, and let $a \in \Sigma$ be such that the language $J_a$ does not contain a string which admits a reduction sequence of length $m$. Then $L_a$ is a regular language, and a finite-state acceptor for $L_a$ can be constructed effectively from $T$, $R$, $a$ and $m$.

**Proof.** If $u_0 a v_0 \in L_a$, then $u_0 a v_0 \in I_a$ and $u_0 a v_0 \rightarrow^*_T u_0$. Since $I_a$ is a regular language, there is a finite-state acceptor $B_a$ for this language. A finite-state acceptor for the language $L_a$ is thus obtained by combining $B_a$ with a finite-state acceptor $C_a$ that is to verify the condition $u_0 a v_0 \rightarrow^*_T u_0$.

By the hypothesis the string $u_0 a v_0 \in J_a$ does not admit a reduction sequence of length $m$. Hence, $u_0$ and $v_0$ can be factored as $u_0 = u_1 u_2$, $v_0 = v_2 v_1$, $|u_2|, |v_2| \leq (m-1) \cdot \mu$, such that $u_2 a v_2 \rightarrow^*_T w$ and $u_1 w v_1 \in \text{IRR}(T)$, i.e., the process of reducing this string to some irreducible descendant actually involves only a factor of length at most $2 \cdot (m-1) \cdot \mu + 1$ surrounding the distinguished occurrence of the letter $a$.

For the construction of $C_a$ we need to distinguish between (i) the case that the string-rewriting system $T$ is monadic and confluent and (ii) the case that $T$ is special and $\lambda$-confluent.

- **Case (i):** If $u_0 \# a v_0 \in L_a$, then $u_0 a v_0 \rightarrow^*_T u_0$, and hence, since $T$ is confluent and $u_0$ is irreducible, $u_0 a v_0 \rightarrow^*_T u_1 w v_1 = u_0 = u_1 u_2$. Thus, $u_0 = u_1 u_2$, $v_0 = v_2 v_1$, $|u_2|, |v_2| \leq (m-1) \cdot \mu$, $u_2 a v_2 \rightarrow^*_T w$ and $w v_1 = u_2$. Hence, we can design the finite-state acceptor $C_a$ to work as follows:

  - On input $u_0 \# a v_0$, $C_a$ reads $u_0$ from left to right always storing the last $(m-1) \cdot \mu$ symbols read in its finite control. Thus, when the symbol $\#$ is encountered, the finite
control contains the suffix \( u_2 \) of \( u_0 \) of length \((m - 1) \cdot \mu \). Then the prefix \( v_2 \) of \( v_0 \) of length \((m - 1) \cdot \mu \) is read and also stored in the finite control. Upon reading the \((m - 1) \cdot \mu\)th symbol of \( v_0 \), the contents of the finite control is replaced by the pair of strings \((w, u_2)\), where \( w \) is the irreducible descendant of \( u_2 a v_2 \). Now \( C_a \) accepts if and only if \( w v_1 = u_2 \), where \( v_1 \) is the remaining input. It is easily seen from the above discussion that a string \( x \) is accepted by both \( B_a \) and \( C_a \) if and only if \( x \) is in the language \( L_a \). This completes the proof of case (i).

Case (ii): If \( u_0 v_0 \in L_a \), then \( u_0 a v_0 \in \mathcal{T} u_0 \). Hence, by Proposition 2.3 \( u_0 a v_0 \) and \( u_0 \) have normal forms \( u_0 a v_0 = x_0 a_1 x_1 \ldots a_r x_r \) and \( u_0 = y_0 a_1 y_1 \ldots a_r y_r \), respectively, such that \( x_i \leftrightarrow \mathcal{T} y_i \), \( i = 0, 1, \ldots, r \). Since \( u_0 \) is a prefix of \( u_0 a v_0 \), we can conclude that there is an index \( s \in \{0, 1, \ldots, r\} \) such that the following properties hold:

\[
\begin{align*}
&x_s = x_0 a_1 y_1 \ldots a_r y_r \triangleright \mathcal{T} x_s, \\
&x_s = x_0 a_1 y_1 \ldots a_r y_r \triangleright \mathcal{T} x_s, \\
&v_2 = y_1 \ldots a_r y_r a x_1' \triangleright \mathcal{T} a y_1' a x_1',
\end{align*}
\]

Recall that the \( y_i \) are maximal invertible factors of \( u_0 \). Thus, if \( s - 1 \) is the largest index such that \( x_i = y_i \) for all \( i = 0, 1, \ldots, s - 1 \), then \( x_s \) cannot be a factor of \( u_0 \), i.e., \( y_s a_1 y_{s+1} \ldots a_r y_r \) is a proper prefix of \( x_s \).

Since \( y_i \) is invertible, the congruence \( y_i \leftrightarrow \mathcal{T} x_s = y_s a_{s+1} y_{s+1} \ldots a_r y_r a x'_i \triangleright \mathcal{T} x_s \), and hence, since \( T \) is \( \lambda \)-confluent, that \( a_{s+1} y_{s+1} \ldots a_r y_r a x'_i \triangleright \mathcal{T} \lambda \). Thus, \( |x'_i|, |a_{s+1} y_{s+1} \ldots a_r y_r| \leq (m - 1) \cdot \mu \). Hence, we can design the finite-state acceptor \( C_a \) to work as follows:

On input \( u_0 \mathcal{S}_a v_0 \), \( C_a \) reads \( u_0 \) from left to right always storing the last \((m - 1) \cdot \mu \) symbols read in its finite control. Thus, when the symbol \( \mathcal{S}_a \) is encountered, the finite control contains the suffix \( u_2 \) of \( u_0 \) of length \((m - 1) \cdot \mu \). On reading the symbol \( \mathcal{S}_a \), the string \( u_2 \) is replaced by the longest suffix \( a_1 y_1 \ldots a_r y_r a x'_s \triangleright \mathcal{T} x_s \), where \( x_2 = x_0 \mathcal{S}_a v_3, a_{s+1} y_{s+1} \ldots a_r y_r a x'_s \triangleright \mathcal{T} x_s \triangleright \mathcal{T} \lambda, y_2 \mathcal{S}_a \mathcal{S}_a \) is a formal inverse of the string \( y_i \), and \( a_i' \) is a specially marked copy of the symbol \( a_i \) \((i = s + 1, \ldots, r)\). It remains to verify that the normal form of \( v_3 \mathcal{S}_a v_1 \) has the form \( a_1 x_1 \ldots a_r x_r \) with \( x_2 \mathcal{S}_a \mathcal{S}_a \), \( i = s + 1, \ldots, r \).

Since \( T \) is \( \lambda \)-confluent, \( x_i \leftrightarrow \mathcal{T} y_i \) if and only if \( y_i \mathcal{S}_a \mathcal{S}_a \), and since \( T \) is a special system and \( v_3 \mathcal{S}_a v_1 \) is irreducible, \( C_a \) can check this while reading \( v_3 \mathcal{S}_a v_1 \) from left to right. Thus, as in case (i) we obtain a finite-state acceptor for the language \( L_a \) by combining \( B_a \) and \( C_a \). This completes the proof of Lemma 3.6. □

Since statement (2) holds for \( T \) and \( R \) if and only if \( L_a \neq \emptyset \) for some \( a \in \Sigma \), and since the emptiness problem for regular languages is decidable, we obtain the following conclusion.

**Corollary 3.7.** The following problem is decidable.
Instance: A finite string-rewriting system $T$ that is either monadic and confluent or that is special and $\lambda$-confluent, a regular set $R \subseteq \Sigma^*$, and an integer $m \in \mathbb{N}$ such that none of the languages $J_a$ ($a \in \Sigma$) contains a string that admits a reduction sequence of length $m$.

Question: Does statement (2) hold for $T$ and $R$?

It remains to deal with statement (3). For $a \in \Sigma$, let $K_a$ and $M_a$ be the following languages:

$$K_a := \{ u_0 \# v_0 \mid u_0, v_0 \in IRR(T), \text{ and } \exists u, v \in \Sigma^*: u \rightarrow^*_T u_0, v \rightarrow^*_T v_0, \text{ and } uav \in R \},$$

and

$$M_a := \{ u_0 \# v_0 \mid u_0, v_0 \in IRR(T), \text{ and } \exists u, v \in \Sigma^*: u \rightarrow^*_T u_0, v \rightarrow^*_T v_0, \text{ and } uav \in R \}. $$

Along the lines of the proofs of Lemma 3.3, Corollary 3.4, and Lemma 3.5 the following can be shown.

**Lemma 3.8.**

(a) The languages $K_a$ and $M_a$ are regular, and from $R$, $T$ and $a \in \Sigma$, finite-state acceptors can be constructed effectively for them.

(b) If, for some $a \in \Sigma$, the language $M_a$ contains a string that admits a reduction sequence of length $(n + 1) \cdot \sigma_R + 1$, then there are two distinct strings $x, y \in R$ with $x \leftrightarrow^*_T y$.

Further, for $a, b \in \Sigma$, $a \neq b$, we consider the language

$$H_{a,b} := \Sigma^* \cdot \{\#\} \cdot (\Sigma_0 \times \Sigma_0)^*,$$

where $\Sigma_0 := \Sigma \cup \{\perp\}$, which is defined as follows:

$$H_{a,b} := \left\{ u_0 \# \left\lfloor \begin{array}{c} v_0 \\ w_0 \end{array} \right\rfloor u_0, v_0, w_0 \in IRR(T), \text{ and } \exists u, v, w \in \Sigma^*: u \rightarrow^*_T u_0, v \rightarrow^*_T v_0, w \rightarrow^*_T w_0, \text{ and } uav, ubw \in R \right\}. $$

Here $\perp$ and $\#$ are new symbols, and $\left\lfloor \begin{array}{c} v_0 \\ w_0 \end{array} \right\rfloor$ stands for $(v_0^m) (b_1^m) \cdots (b_s^m)$, where $v_0 = a_0 a_1 \ldots a_r \in \Sigma^*$, $w_0 = b_0 b_1 \ldots b_s \in \Sigma^*$, $m = \max \{r, s\}$, and $a_i = \perp$ and $b_j = \perp$ for $i > r$ and $j > s$, respectively. If $u_0 \# \left\lfloor \begin{array}{c} v_0 \\ w_0 \end{array} \right\rfloor \in H_{a,b}$, then there exist strings $uav \in R$ and $ubw \in R$ such that $u \rightarrow^*_T u_0$, $v \rightarrow^*_T v_0$, and $w \rightarrow^*_T w_0$. Now we define $L_{a,b} \subseteq \Sigma^* \cdot \{\#\} \cdot (\Sigma_0 \times \Sigma_0)^*$ through

$$L_{a,b} := \left\{ u_0 \# \left\lfloor \begin{array}{c} v_0 \\ w_0 \end{array} \right\rfloor \in H_{a,b} \mid u_0 av_0 \leftrightarrow^*_T u_0 bw_0 \right\}. $$
If \( u_0 \# [\omega_0] \in L_{a,b} \), then there exist strings \( uav \in R \) and \( ubw \in R \) such that \( uav \mapsto^* \) \( u_0av_0 \mapsto^* u_0bw_0 \mapsto^* ubw \), i.e., statement (3) is satisfied for \( T \) and \( R \). Conversely, if this statement is satisfied for \( T \) and \( R \), then, for some letters \( a, b \in \Sigma \), \( a \neq b \), \( L_{a,b} \neq \emptyset \). Thus, we see that statement (3) is satisfied for \( T \) and \( R \) if and only if \( L_{a,b} \) is nonempty for some letters \( a, b \in \Sigma \), \( a \neq b \). Therefore, the following technical results are of interest.

**Lemma 3.9.** Let \( T \) be a finite monadic and confluent string-rewriting system, and let \( a, b \in \Sigma \) such that \( a \neq b \).

(a) The language \( H_{a,b} \) is regular, and from \( R, T \) and \( a, b \in \Sigma \), a finite-state acceptor can be constructed effectively for it.

(b) Let \( m \in \mathbb{N} \) be such that neither \( M_a \) nor \( M_b \) contains a string that admits a reduction sequence of length \( m \). Then the language \( L_{a,b} \) is regular, and a finite-state acceptor for \( L_{a,b} \) can be constructed from \( T, R, a, b \) and \( m \).

**Proof.** (a) Consider the language \( R_{a,b} = \{ u \# v \# w \mid uav \in R \) and \( ubw \in R \} \). This language is certainly regular. In fact, a finite-state acceptor for \( R_{a,b} \) can be constructed from the product of two copies of a finite-state acceptor for \( R \). This product acceptor would work in 3 phases: In phase 1, while the factor \( u \) is being read, both copies would work in parallel. In phase 2, while the factor \( v \) is being read, one copy would process the input, while the other would be idle. Finally, in phase 3, while the factor \( w \) is being read, the first copy would be idle, while the other would process the input. Using the construction described in [2] we can then obtain a finite-state acceptor for the language \( \Delta^*_T(R_{a,b}) \cap \text{IRR}(T) \cdot \{ \} \cdot \text{IRR}(T) \cdot \{ \} \cdot \text{IRR}(T) = \{ u_0 \# v_0 \# w_0 \mid u_0, v_0, \ w_0 \in \text{IRR}(T), \) and \( uav, ubw \in R : u \mapsto^* u_0, v \mapsto^* v_0, \) and \( w \mapsto^* w_0 \} \). This finite-state acceptor will essentially still work in 3 phases. Now by running phases 2 and 3 in parallel, we obtain a finite-state acceptor for the language \( H_{a,b} \).

(b) If \( u_0 \# [\omega_0] \in L_{a,b} \), then \( u_0 \# [\omega_0] \in H_{a,b} \), and \( u_0av_0 \mapsto^* u_0bw_0 \). Thus, we obtain a finite-state acceptor for the language \( L_{a,b} \) by combining the acceptor for \( H_{a,b} \) with a finite-state acceptor \( C_{a,b} \) that is to verify the condition \( u_0av_0 \mapsto^* u_0bw_0 \).

If \( u_0 \# [\omega_0] \in L_{a,b} \), then \( u_0av_0 \in M_a \) and \( u_0bw_0 \in M_b \). By our hypothesis neither the string \( u_0av_0 \in M_a \) nor the string \( u_0bw_0 \in M_b \) admits a reduction sequence of length \( m \). Thus, if \( u_0av_0 \mapsto^*_T u_0bw_0 \), then \( u_0, v_0, w_0 \) have factorizations of the form \( u_0 = u_1u_2, v_0 = v_2v_1, \) and \( w_0 = w_3w_1 \) such that \( |u_2|, |v_2|, |w_2| \leq (m-1) \cdot \mu, u_2av_2 \mapsto^*_T g, u_2bw_2 \mapsto^*_T h \) and \( u_1gv_1 = u_1hw_1 \in \text{IRR}(T) \). Hence, we can design the finite-state acceptor \( C_{a,b} \) to work as follows:

On input \( u_0 \# [\omega_0] \), \( C_{a,b} \) reads \( u_0 \) from left to right always remembering the last \( (m-1) \cdot \mu \) symbols read. Then the prefix \( [\omega_i'] \) of \( [\omega_0'] \) of length \( (m-1) \cdot \mu \) is read, and upon reading the last symbol of \( [\omega_i'] \), the pair of strings \((g, h)\) is stored in \( C_{a,b}'s \) finite control. Now \( C_{a,b} \) accepts if and only if \( gv_1 = hw_1 \), where \( [\omega_i'] \) is the remaining input. \( \square \)
To deal with the case that the string-rewriting system $T$ is special and $\lambda$-confluent, we consider the following languages $H_{a,b}$ and $K_{a,b}$ ($a, b \in \Sigma, a \neq b$):

$$H_{a,b} = \{ \rho(v_0)\$_a w_0 \mid u_0, v_0, w_0 \in \text{IRR}(T), \text{ and} \}$$

$$\exists u, v, w \in \Sigma^*: u \rightarrow _T u_0, v \rightarrow _T v_0, w \rightarrow _T w_0, \text{ and } uav, ubw \in R \},$$

where $\$_a$ is a new letter, and $\rho$ denotes the function reversal, and

$$K_{a,b} = \{ \rho(v_0)\$_a w_0 \in H_{a,b} \mid u_0av_0 \leftrightarrow _T u_0bw_0 \}.$$ 

If $\rho(v_0)\$_a w_0 \in K_{a,b}$, then there are strings $u, v, w \in \Sigma^*$ such that $u \rightarrow _T u_0, v \rightarrow _T v_0, w \rightarrow _T w_0, uav \in R, ubw \in R$, and $uav \leftrightarrow _T ubw$. Hence, statement (3) is satisfied for $T$ and $R$ if and only if $K_{a,b}$ is nonempty for some letters $a, b \in \Sigma, a \neq b$. We want to prove that under certain conditions the language $K_{a,b}$ is deterministic context-free.

**Lemma 3.10.** Let $T$ be a finite special and $\lambda$-confluent string-rewriting system, let $a, b \in \Sigma, a \neq b$, and let $m \in \mathbb{N}$ be such that neither $M_a$ nor $M_b$ contains a string that admits a reduction sequence of length $m$. Then the language $K_{a,b}$ is deterministic context-free, and a deterministic pushdown automaton (dpda) accepting this language can be constructed from $T, R, a, b$ and $m$.

**Proof.** As observed in the proof of Lemma 3.9(a) the language $\{ u_0 \# v_0 \# w_0 \mid u_0, v_0, w_0 \in \text{IRR}(T), \text{ and } 3uav, ubw \in R: u \rightarrow _T u_0, v \rightarrow _T v_0, w \rightarrow _T w_0 \}$ is regular, and a finite-state acceptor for it can be constructed effectively. From this it is fairly easy to see that the language $H_{a,b}$ is regular, and that a (deterministic) finite-state acceptor for it can be constructed effectively. From this it is fairly easy to see that the language $H_{a,b}$ is regular, and that a (deterministic) finite-state acceptor for $H_{a,b}$ can be obtained. A dpda for the language $K_{a,b}$ is now obtained by combining the finite-state acceptor for $H_{a,b}$ with a dpda $P_{a,b}$ that is to check the condition $u_0av_0 \leftrightarrow _T u_0bw_0$. We design the dpda $P_{a,b}$ to work as follows:

On input $\rho(v_0)\$_a w_0$, $P_{a,b}$ first reads the prefix $\rho(v_0)$. While doing this $P_{a,b}$ computes the normal form $x_0a_1x_1 \cdots a_rx_r$ of $v_0$, pushing the string $y_1b_1 \cdots y_rb_r$ onto its stack. Here $y_i$ denotes an irreducible descendant of the inverse $x_i^{-1}$ of the string $x_i$ ($i = 0, 1, \ldots, r$), and $\Gamma = \{ b_i \mid a_i \in \Sigma \}$ is a new alphabet in one-to-one correspondence to $\Sigma$. (Recall Proposition 2.3 and the discussion following it.) Thus, $P_{a,b}$ stores the normal form of an irreducible string presenting the 'inverse' of the string $v_0$ on its stack. Observe that this does not cause any problems, since $P_{a,b}$ is reading the reversal $\rho(v_0)$ of the string $v_0$, and the gsm $G_{NF}$ can be incorporated in the finite control of $P_{a,b}$. In addition, the prefix $v_2$ of $v_0$ of length $(m - 1)$ is stored in $P_{a,b}$'s finite control.

Then the factor $u_0$ is read, and its suffix $u_2$ of length $(m - 1)$ is stored in the finite control. Finally, the prefix $w_2$ of $w_0$ of length $(m - 1)$ is read into $P_{a,b}$'s finite control. Now within its finite control $P_{a,b}$ performs the two reductions $u_2av_2 \rightarrow _T g \in \text{IRR}(T)$ and $u_2bw_2 \rightarrow _T h \in \text{IRR}(T)$. Since neither $M_a$ nor $M_b$ contains a string that admits a reduction sequence of length $m$, the strings $u_1gv_1$ and $u_1hw_1$ are irreducible, where $u_0 = u_1u_2$, $v_0 = v_2v_1$, and $w_0 = w_2w_1$. Thus, $u_0av_0 \leftrightarrow _T u_0bw_0$ if and only if $u_1gv_1 \leftrightarrow _T u_1hw_1$ if and only if the normal forms of $u_1gv_1$ and of $u_1hw_1$ are related to
each other as expressed by Proposition 2.3(b). Checking this essentially amounts to comparing the normal form of $gv_1$ to the normal form of $hw_1$. Because the stack already contains the normal form of the 'inverse' of $v_0$, this is easily done by reduction, since the system $T$ is special and $\lambda$-confluent. This completes the proof of Lemma 3.10. 

Since statement (3) holds for $T$ and $R$ if and only if $L_{a,b} \neq \emptyset$ for some $a, b \in \Sigma$, $a \neq b$, respectively if $K_{a,b} \neq \emptyset$, we have the following conclusion.

**Corollary 3.11.** The following problem is decidable.

**Instance:** A finite string-rewriting system $T$ that is either monadic and confluent or that is special and $\lambda$-confluent, a regular set $R \subseteq \Sigma^*$, and an integer $m \in \mathbb{N}$ such that none of the languages $M_a (a \in \Sigma)$ contains a string that admits a reduction sequence of length $m$.

**Question:** Does statement (3) hold for $T$ and $R$?

We need one additional technical result.

**Lemma 3.12.** Let $T$ be a finite monadic string-rewriting system, and let $R \subseteq \Sigma^*$ be a regular set that is specified through some finite-state acceptor. Then the set $\Delta(R) = \{ y \in \Sigma^* | \exists x \in R : x \rightarrow^*_T y \}$ is regular, and a finite-state acceptor for $\Delta(R)$ can be constructed effectively.

**Proof.** Obviously, $y \in \Delta(R)$ if and only if there is a rule $(l \rightarrow r) \in T$ such that $y = y_1 r y_2$ and $y_1, y_2 \in R$. Hence, a finite-state acceptor for $\Delta(R)$ can easily be obtained from $T$ and a finite-state acceptor for $R$. 

Now we can combine our technical results to get a proof for Theorem 3.1. Let $T$ be a finite string-rewriting system that is either monadic and confluent or special and $\lambda$-confluent, let $n = |\Sigma|$, and let $R \subseteq \Sigma^*$ be a regular set. First, the integer $m \geq (n + 1) \cdot \sigma_R + 1$ is computed. Then, for each $a \in \Sigma$, a finite-state acceptor for the language $J_a$ is constructed. Now, for $a \in \Sigma$, $J_a$ contains a string that admits a reduction sequence of length $m$ if and only if $A^m(J_a)$ is nonempty. By Lemma 3.12 this can be checked for all $a \in \Sigma$. If, for some $a \in \Sigma$, $J_a$ does contain such a string, then by Lemma 3.5 there exist strings $x, y \in R$ such that $x \neq y$ and $x \leftrightarrow^*_T y$. Otherwise, by Corollary 3.7 we can verify whether or not statement (2) holds for $T$ and $R$. In the affirmative, there are distinct strings $x, y \in R$ with $x \leftrightarrow^*_T y$. Otherwise, we construct finite-state acceptors for the languages $M_a (a \in \Sigma)$, and check whether, for some $a \in \Sigma$, $M_a$ contains a string that admits a reduction sequence of length $m$. Again, this holds if and only if $A^m(M_a) \neq \emptyset$. In the affirmative, $R$ contains distinct strings $x, y$ with $x \leftrightarrow^*_T y$ by Lemma 3.8(b); otherwise, we can decide whether statement (3) holds for $T$ and $R$ by Corollary 3.11. In this situation $R$ contains distinct strings $x, y$ with $x \leftrightarrow^*_T y$ if and only if statement (3) holds for $T$ and $R$, which completes the proof of Theorem 3.1, and therewith of Theorem 2.2.
References