Reachability Analysis with State-Compatible Automata*

Bertram Felgenhauer and René Thiemann

Institute of Computer Science, University of Innsbruck, Innsbruck, Austria
{bertram.felgenhauer,rene.thiemann}@uibk.ac.at

Abstract. Regular tree languages are a popular device for reachability analysis over term rewrite systems, with many applications like analysis of cryptographic protocols, or confluence and termination analysis. At the heart of this approach lies tree automata completion, first introduced by Genet for left-linear rewrite systems. Korp and Middeldorp introduced so-called quasi-deterministic automata to extend the technique to non-left-linear systems. In this paper, we introduce the simpler notion of state-compatible automata, which are slightly more general than quasi-deterministic, compatible automata. This notion also allows us to decide whether a regular tree language is closed under rewriting, a problem which was not known to be decidable before.
Several of our results have been formalized in the theorem prover Isabelle/HOL. This allows to certify automatically generated non-confluence and termination proofs that are using tree automata techniques.

1 Introduction

In this paper we are largely concerned with over-approximations of the terms reachable from a regular tree language L_0 by rewriting using a term rewrite system $\mathcal{R}$, that is, we are interested in regular tree languages L such that $\mathcal{R}^*(L_0) \subseteq L$. Such over-approximations have been used, among other things, in the analysis of cryptographic protocols [6], for termination analysis [7,10] and for establishing non-confluence of term rewrite systems [15].

Unfortunately, the question whether $\mathcal{R}^*(L_0) \subseteq L$ is undecidable in general. Tree automata completion, conceived by Genet et al. [4,5], is based on the stronger requirements that $L_0 \subseteq L$ and L is itself closed under rewriting, i.e., $\mathcal{R}(L) \subseteq L$. This is accomplished by constructing L as the language accepted by a bottom-up tree automaton $\mathcal{A}$ that is *compatible* with $\mathcal{R}$: Whenever $l\sigma$ is accepted in state q by $\mathcal{A}$, where $l \to r \in \mathcal{R}$ and σ maps variables to states of $\mathcal{A}$, we demand that $r\sigma$ is also accepted in q. If $\mathcal{A}$ is deterministic or if $\mathcal{R}$ is a left-linear term rewrite system, then compatibility ensures that $\mathcal{L}(\mathcal{A})$ is closed under rewriting by $\mathcal{R}$.

Example 1. Let $\mathcal{R} = \{\mathsf{f}(x, x) \to x\}$ and $\mathcal{A}$ be the automaton with states $1, 2, 3$, final state 3, and transitions

$$\mathsf{a} \to 1 \qquad \mathsf{a} \to 2 \qquad \mathsf{f}(1, 2) \to 3$$

* This research was supported by FWF projects P22467 and P22767.

So $\mathcal{A}$ is non-deterministic and $\mathcal{R}$ is non-left-linear. Even though $\mathcal{A}$ is compatible with $\mathcal{R}$, $\mathcal{L}(\mathcal{A}) = \{\mathsf{f}(\mathsf{a},\mathsf{a})\}$ is not closed under rewriting by $\mathcal{R}$, because $\mathsf{f}(\mathsf{a},\mathsf{a})$ can be rewritten to a which is not in $\mathcal{L}(\mathcal{A})$.

However, demanding $\mathcal{A}$ to be deterministic if $\mathcal{R}$ is not left-linear may result in bad approximations.

Example 2. Let $\mathcal{R} = \{\mathsf{f}(x,x) \to \mathsf{b}, \mathsf{b} \to \mathsf{a}\}$ and $L_0 = \{\mathsf{f}(\mathsf{a},\mathsf{a})\}$. The set of terms reachable from L_0, namely $\mathcal{R}^*(L_0) = \{\mathsf{f}(\mathsf{a},\mathsf{a}), \mathsf{b}, \mathsf{a}\}$, is not accepted by any deterministic, compatible tree automaton. To see why, assume that such an automaton $\mathcal{A}$ exists, and let q be the state accepting $\mathsf{f}(\mathsf{a},\mathsf{a})$. There must be transitions $\mathsf{a} \to q'$ (q' is unique because $\mathcal{A}$ is deterministic) and $\mathsf{f}(q',q') \to q$ in $\mathcal{A}$. By compatibility with the rules $\mathsf{f}(x,x) \to \mathsf{b}$ and $\mathsf{b} \to \mathsf{a}$, we must have transitions $\mathsf{b} \to q$, and $\mathsf{a} \to q$. Since we already have the transition $\mathsf{a} \to q'$, determinism implies $q' = q$. With the three transitions $\mathsf{a} \to q$, $\mathsf{b} \to q$, and $\mathsf{f}(q,q) \to q$, $\mathcal{A}$ accepts every term over the signature $\{\mathsf{f},\mathsf{a},\mathsf{b}\}$, which is not a very useful approximation of $\mathcal{R}^*(L_0)$.

To overcome this problem, Korp and Middeldorp introduced quasi-deterministic automata [10]. Indeed it is easy to find a quasi-deterministic automaton accepting $\mathcal{R}^*(L_0) = \{\mathsf{f}(\mathsf{a},\mathsf{a}), \mathsf{b}, \mathsf{a}\}$ that is compatible with $\mathcal{R}$ from the previous example.

Example 3. Let $\mathcal{A}$ be an automaton with states $1, 2$, final state 2 and transitions

$$\mathsf{a} \to 1^* \qquad \mathsf{a} \to 2 \qquad \mathsf{b} \to 2^* \qquad \mathsf{f}(1,1) \to 2^*$$

where the stars indicate the so-called designated states for each left-hand side. Then $\mathcal{A}$ is quasi-deterministic, compatible with $\mathcal{R}$ and $\mathcal{L}(\mathcal{A}) = \{\mathsf{f}(\mathsf{a},\mathsf{a}), \mathsf{b}, \mathsf{a}\}$.

In this paper, we concentrate on the compatibility requirement that ensures $\mathcal{R}(L) \subseteq L$. Since there may be bugs in the implementation of tree automata completion, it is important to independently certify whether $\mathcal{R}(L) \subseteq L$ is really satisfied. Such a certifier has already been developed in [2], but it is restricted to left-linear systems and does not support the stronger quasi-deterministic automata. We extend this work by introducing *state-compatible* automata, which are deterministic but accomplish the effect of quasi-deterministic automata by relaxing the compatibility requirement instead. It turns out that as long as $\mathcal{R}$ has only non-collapsing rules, state-compatible automata and quasi-deterministic automata are equivalent. In the presence of collapsing rules, state-compatible automata can capture more approximations than quasi-deterministic ones.

We will further show that state-compatibility does not only ensure $\mathcal{R}(L) \subseteq L$, but it can also be utilized to obtain a decision procedure for the question whether a regular tree language is closed under rewriting—a problem whose decidability was hitherto unknown, as far as we know. These results have also been formalized within the theorem prover Isabelle/HOL [13], resulting in a formalized decision procedure for the question $\mathcal{R}(L) \subseteq L$. It is used to certify non-confluence proofs and termination proofs that are using the techniques of [9,10,15].

This paper is structured as follows. In Section 2 we recall basic definitions and notation. The main part of our paper is Section 3, where we introduce the notions of state-coherence and state-compatibility, and present the decision procedure. Section 4 is devoted to a comparison to quasi-deterministic automata. Details on the formalization are provided in Section 5. Finally, we conclude in Section 6.

2 Preliminaries

We assume that the reader is familiar with first order term rewriting and tree automata. For introductions to these topics see [1] and [3].

Terms over a signature $\mathcal{F}$ and a set of variables $\mathcal{V}$, denoted $\mathcal{T}(\mathcal{F},\mathcal{V})$ (or $\mathcal{T}(\mathcal{F})$ if $\mathcal{V}$ is empty) are inductively defined as either variables $v \in \mathcal{V}$ or of the form $f(t_1,\dots,t_n)$, where $t_1,\dots,t_n$ are terms and $f \in \mathcal{F}$ is a function symbol of arity n. We write $\mathcal{V}\mathsf{ar}(t)$ for the set of variables in t. A term t is linear if each variable occurs in t at most once. Contexts are terms over $\mathcal{F} \cup \{\Box\}$ that contain exactly one occurrence of $\Box$. If C is a context and t a term, then $C[t]$ denotes the term obtained by replacing the $\Box$ in C by t. A substitution $\sigma : \mathcal{V} \to \mathcal{T}(\mathcal{F},\mathcal{V})$ maps variables to terms. We write $t\sigma$ for the result of replacing each variable x in t by $\sigma(x)$.

A term rewrite system (TRS) $\mathcal{R}$ is a set of rewrite rules $l \to r$, where each rule's left-hand side l and right-hand side r are terms such that $l \notin \mathcal{V}$ and $\mathcal{V}\mathsf{ar}(r) \subseteq \mathcal{V}\mathsf{ar}(l)$. A TRS $\mathcal{R}$ defines a rewrite relation $\to_{\mathcal{R}}$, namely $s \to_{\mathcal{R}} t$ whenever there are a context C, a rule $l \to r \in \mathcal{R}$, and a substitution σ such that $s = C[l\sigma]$ and $t = C[r\sigma]$. We denote by $\mathrm{lhs}(\mathcal{R})$ the set of all left-hand sides of rules in $\mathcal{R}$. A TRS is left-linear if all its left-hand sides are linear terms. A rule $l \to r$ is called collapsing if r is a variable. The inverse, the reflexive closure, transitive closure, and the reflexive, transitive closure of a binary relation $\to$ are denoted by $\leftarrow$, $\to^=$, $\to^+$, and $\to^*$, respectively. Given a set of terms L, $\mathcal{R}(L)$ ($\mathcal{R}^*(L)$) is the set of one-step (many-step) descendants of L: $t' \in \mathcal{R}(L)$ ($t' \in \mathcal{R}^*(L)$) iff $t \to_{\mathcal{R}} t'$ ($t \to^*_{\mathcal{R}} t'$) for some $t \in L$. A language L is closed under rewriting by $\mathcal{R}$, if $\mathcal{R}(L) \subseteq L$.

A (bottom-up) tree automaton $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ over a signature $\mathcal{F}$ consists of a set of states Q disjoint from $\mathcal{F}$, a set of final states $Q_f \subseteq Q$, and a set of transitions Δ of shape $f(q_1,\dots,q_n) \to q$ where the root $f \in \mathcal{F}$ has arity n and $q, q_1, \dots, q_n \in Q$. (We forbid ε-transitions for the sake of simplicity.) We regard Δ as a TRS over the signature $\mathcal{F} \cup Q$, with the states as constants. A substitution σ is a state substitution if $\sigma(x) \in Q$ for all $x \in \mathcal{V}$. A term t is accepted in state q if $t \to^*_\Delta q$; t is accepted by $\mathcal{A}$ if it is accepted in a final state. The language accepted by $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \{t \mid t \to^*_\Delta q \text{ for some } q \in Q_f\}$. We call $\mathcal{A}$ deterministic if no two rules in Δ have the same left-hand side. For convenience, we often write $\to_{\mathcal{A}}$ for $\to_\Delta$. Following [10], we formulate Genet's result from [5] as follows:

Definition 4. *A tree automaton $\mathcal{A}$ is* compatible *with a TRS $\mathcal{R}$ if for all state substitutions σ, rules $l \to r \in \mathcal{R}$ and states $q \in Q$, $l\sigma \to^*_{\mathcal{A}} q$ implies $r\sigma \to^*_{\mathcal{A}} q$.*

Theorem 5. *Let the tree automaton $\mathcal{A}$ be compatible with the TRS $\mathcal{R}$. Then*

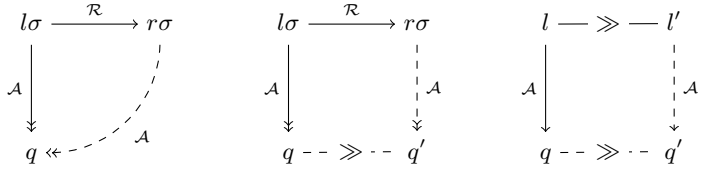

Fig. 1. compatibility, state-compatibility, and state-coherence

1. *if $\mathcal{R}$ is left-linear, then $\mathcal{L}(\mathcal{A})$ is closed under rewriting by $\mathcal{R}$, and*
2. *if $\mathcal{A}$ is deterministic, then $\mathcal{L}(\mathcal{A})$ is closed under rewriting by $\mathcal{R}$.*

Finally, we recall that every tree automaton can be reduced to an equivalent automaton where all states are useful.

Definition 6. *Let $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ be a tree automaton. We say that a state $q \in Q$ is* reachable *if $t \to_{\mathcal{A}}^* q$ for some term $t \in \mathcal{T}(\mathcal{F})$; $q \in Q$ is* productive *if $C[q] \to_{\mathcal{A}}^* q'$ for some context C and state $q' \in Q_f$. Finally, an automaton $\mathcal{A}$ is* trim *if all its states are both reachable and productive.*

Proposition 7. *For any tree automaton $\mathcal{A}$ there is an equivalent tree automaton $\mathcal{A}'$ that is trim. If $\mathcal{A}$ is deterministic, then $\mathcal{A}'$ is also deterministic.*

3 State-Compatible Automata

3.1 Definitions

Before we get down to definitions, let us briefly analyze the failure in Example 2. What happens there is that, by the compatibility requirement, all three terms in the rewrite sequence $\mathsf{f}(\mathsf{a},\mathsf{a}) \to_{\mathcal{R}} \mathsf{b} \to_{\mathcal{R}} \mathsf{a}$ have to be accepted in the same state. In conjunction with the determinism requirement, this is fatal. Consequently, because our goal is to obtain a deterministic automaton, we must allow a and b to be accepted in separate states, q_{a} and q_{b}. To track their connection by rewriting, we introduce a relation $\gg$ on states, such that $q_{\mathsf{b}} \gg q_{\mathsf{a}}$. In general, we require $\gg$ to be state-compatible and state-coherent, which are defined as follows (see also Figure 1).

Definition 8. *Let $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ be a tree automaton, and $\gg \subseteq Q \times Q$ be a relation on the states of $\mathcal{A}$. We say that $(\mathcal{A}, \gg)$ is* state-compatible with a TRS *$\mathcal{R}$ if for all state substitutions σ, rules $l \to r \in \mathcal{R}$ and states $q \in Q$, if $l\sigma \to_{\mathcal{A}}^* q$ then $r\sigma \to_{\mathcal{A}}^* q'$ for some $q' \in Q$ with $q \gg q'$. We say that $(\mathcal{A}, \gg)$ is* state-coherent *if $\{q' \mid q \in Q_f, q \gg q'\} \subseteq Q_f$, and if for all $f(q_1, \ldots, q_i, \ldots, q_n) \to q \in \Delta$ and $q_i \gg q_i'$ there is some $q' \in Q$ with $f(q_1, \ldots, q_i', \ldots, q_n) \to q' \in \Delta$ and $q \gg q'$.*

The purpose of state-coherence is to deal with contexts in rewrite steps, as we will see in the proof of Theorem 11 below.

Example 9. Let $\mathcal{A}$ be an automaton with states $1, 2$ (both final), and transitions

$$\mathsf{a} \to 1 \qquad \mathsf{b} \to 2 \qquad \mathsf{f}(1,1) \to 2$$

Furthermore, let $2 \gg 2$ and $2 \gg 1$. Then $(\mathcal{A}, \gg)$ is state-coherent and state-compatible with $\mathcal{R} = \{\mathsf{f}(x,x) \to \mathsf{b}, \mathsf{b} \to \mathsf{a}\}$ and $\mathcal{L}(\mathcal{A}) = \{\mathsf{f}(\mathsf{a},\mathsf{a}), \mathsf{b}, \mathsf{a}\}$. Note that this automaton was obtained from the quasi-deterministic automaton from Example 3 by keeping only the transitions to designated states. We will see in Section 4 that this construction works in general.

Remark 10. If $(\mathcal{A}, \gg)$ is state-coherent, then $(\mathcal{A}, \gg^{=})$ and $(\mathcal{A}, \gg^{*})$ are also state-coherent. The same holds for state-compatibility with $\mathcal{R}$.

3.2 Soundness and Completeness

Next we prove the analogue of Theorem 5 for state-coherent, state-compatible automata.

Theorem 11. *Let $\mathcal{A}$ be a tree automaton such that $(\mathcal{A}, \gg)$ is state-coherent and state-compatible with the TRS $\mathcal{R}$ for some relation $\gg$. Then*

1. *if $\mathcal{R}$ is left-linear, then $\mathcal{L}(\mathcal{A})$ is closed under rewriting by $\mathcal{R}$, and*
2. *if $\mathcal{A}$ is deterministic, then $\mathcal{L}(\mathcal{A})$ is closed under rewriting by $\mathcal{R}$.*

Proof. Let $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$. First we show that whenever $l\tau \to_{\mathcal{A}}^* q$ for some substitution τ and rule $l \to r \in \mathcal{R}$, then there is a state $q' \in Q$ with $q \gg q'$ and $r\tau \to_{\mathcal{A}}^* q'$. By the assumptions, we can extract from $l\tau \to_{\mathcal{A}}^* q$ a state substitution σ such that $l\tau \to_{\mathcal{A}}^* l\sigma \to_{\mathcal{A}}^* q$: For each $x \in \mathcal{V}\mathsf{ar}(l)$, we map x to the state reached from $\tau(x)$ in the given sequence. The state is unique either by left-linearity, or because the given automaton is deterministic. By state-compatibility, we obtain a state q' such that $q \gg q'$ and $r\tau \to_{\mathcal{A}}^* r\sigma \to_{\mathcal{A}}^* q'$.

Using state-coherence we can show by structural induction on C that whenever $C[q] \to_{\mathcal{A}}^* q_\bullet$ and $q \gg q'$, then $C[q'] \to_{\mathcal{A}}^* q'_\bullet$ for some state $q'_\bullet$ with $q_\bullet \gg q'_\bullet$.

Finally, assume that $t \in \mathcal{L}(\mathcal{A})$ and $t \to_{\mathcal{R}} t'$. Then there exist a rule $l \to r \in \mathcal{R}$, a context C and a substitution τ such that $t = C[l\tau]$ and $t' = C[r\tau]$. We have a derivation $t = C[l\tau] \to_{\mathcal{A}}^* C[q] \to_{\mathcal{A}}^* q_\bullet \in Q_f$. By the preceding observations we can find states $q \gg q'$ and $q_\bullet \gg q'_\bullet$ such that $t' = C[r\tau] \to_{\mathcal{A}}^* C[q'] \to_{\mathcal{A}}^* q'_\bullet$. Note that by state-coherence, $q_\bullet \in Q_f$ implies $q'_\bullet \in Q_f$, so that $t' \in \mathcal{L}(\mathcal{A})$. □

Note that Theorem 11 generalizes Theorem 5 (choose $\gg$ to be the identity relation on states, which is always state-coherent). Moreover, the converse of Theorem 11 holds for trim, deterministic automata. We will prove this in Theorem 13 below, which allows us to derive our main decidability result in Corollary 14. But first let us show by example that the converse fails for some trim, non-deterministic automaton and ground TRS $\mathcal{R}$.

Example 12. Consider the TRS $\mathcal{R} = \{\mathsf{a} \to \mathsf{b}\}$ and the automaton $\mathcal{A}$ with states $0, 1, 2, 3$, final state 0, and transitions

$$\begin{array}{llll} \mathsf{a} \to 1 & \mathsf{f}(1) \to 0 & \mathsf{g}(1) \to 0 & \\ \mathsf{b} \to 2 & \mathsf{f}(2) \to 0 & \mathsf{b} \to 3 & \mathsf{g}(3) \to 0 \end{array}$$

This automaton accepts $\mathcal{L}(\mathcal{A}) = \{\mathsf{f}(\mathsf{a}), \mathsf{f}(\mathsf{b}), \mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{b})\}$, which is closed under rewriting by $\mathcal{R}$. Assume that $(\mathcal{A}, \gg)$ is state-coherent and state-compatible with $\mathcal{R}$. By state-compatibility, $\mathsf{a} \to \mathsf{b}$ begets $1 \gg 2$ or $1 \gg 3$. If $1 \gg 2$, then state-coherence, considering the transition $\mathsf{g}(1) \to 0$, requires a transition with left-hand side $\mathsf{g}(2)$, which does not exist. Similarly, if $1 \gg 3$, then $\mathsf{f}(1) \to 0$ requires a transition with left-hand side $\mathsf{f}(3)$, which does not exist.

Theorem 13. *Let $\mathcal{A}$ be a trim, deterministic tree automaton such that $\mathcal{L}(\mathcal{A})$ is closed under rewriting by the TRS $\mathcal{R}$. Then there is a relation $\gg$ such that $(\mathcal{A}, \gg)$ is state-coherent and state-compatible with $\mathcal{R}$.*

Proof. Let $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$. We define $\gg$ as follows: $q \gg q'$ iff for some terms $t, t' \in \mathcal{T}(\mathcal{F})$, we have

$$q \underset{\mathcal{A}}{\overset{*}{\leftarrow}} t \underset{\mathcal{R}}{\longrightarrow} t' \underset{\mathcal{A}}{\overset{*}{\rightarrow}} q' \tag{1}$$

Note that by virtue of $\mathcal{A}$ being deterministic, t and t' determine q and q' uniquely. We show that $(\mathcal{A}, \gg)$ is state-coherent and state-compatible.

1. (state-coherence) If $q \in Q_f$ and $q \gg q'$, then there exist terms t, t' satisfying (1). In particular, $q \in Q_f$ implies $t \in \mathcal{L}(\mathcal{A})$, and $t \to_{\mathcal{R}} t'$ implies $t' \in \mathcal{L}(\mathcal{A})$, because $\mathcal{L}(\mathcal{A})$ is closed under rewriting by $\mathcal{R}$. Because $\mathcal{A}$ is deterministic, t' determines q' uniquely, and $q' \in Q_f$ follows.
2. (state-coherence) Assume that $f(q_1, \ldots, q_n) \to q \in \Delta$ and $q_i \gg q_i'$ for some index i and state q_i'. By (1) there are t_i, t_i' such that $q_i \ {}^*_{\mathcal{A}}\!\leftarrow t_i \to_{\mathcal{R}} t_i' \to^*_{\mathcal{A}} q_i'$. Because all q_j are reachable, we can fix terms t_j with $t_j \to^*_{\mathcal{A}} q_j$ for $j \neq i$. The state q is productive, so there is a context C such that $C[q] \to^*_{\mathcal{A}} q_\bullet \in Q_f$. Let $t = f(t_1, \ldots, t_n)$ and $t' = f(t_1, \ldots, t_i', \ldots, t_n)$. Then $C[t] \in \mathcal{L}(\mathcal{A})$ and $C[t] \to_{\mathcal{R}} C[t']$, hence $C[t'] \in \mathcal{L}(\mathcal{A})$ as well. Consequently, there are states $q', q_\bullet'$ such that

$$C[q] \underset{\mathcal{A}}{\overset{*}{\leftarrow}} C[t] \underset{\mathcal{R}}{\longrightarrow} C[t'] \underset{\mathcal{A}}{\overset{*}{\rightarrow}} C[f(q_1, \ldots, q_i', \ldots, q_n)] \underset{\mathcal{A}}{\longrightarrow} C[q'] \underset{\mathcal{A}}{\overset{*}{\rightarrow}} q_\bullet' \in Q_f$$

   In particular, we have a transition $f(q_1, \ldots, q_i', \ldots, q_n) \to q' \in \Delta$, and $q \gg q'$.
3. (state-compatibility) Assume that $l\sigma \to^*_{\mathcal{A}} q$ for a state substitution σ. All states of $\mathcal{A}$ are reachable, so there is a substitution $\tau : \mathcal{V} \to \mathcal{T}(\mathcal{F})$ with $\tau(x) \to^*_{\mathcal{A}} \sigma(x)$ for all $x \in \mathcal{V}$. Furthermore, q is productive, so that for some context C, $C[q] \to^*_{\mathcal{A}} q_\bullet \in Q_f$. We have $C[l\tau] \in \mathcal{L}(\mathcal{A})$ and $C[l\tau] \to_{\mathcal{R}} C[r\tau]$. Consequently, $C[r\tau] \in \mathcal{L}(\mathcal{A})$ and for some states $q', q_\bullet'$,

$$C[q] \underset{\mathcal{A}}{\overset{*}{\leftarrow}} C[l\sigma] \underset{\mathcal{A}}{\overset{*}{\leftarrow}} C[l\tau] \underset{\mathcal{R}}{\longrightarrow} C[r\tau] \underset{\mathcal{A}}{\overset{*}{\rightarrow}} C[q'] \underset{\mathcal{A}}{\overset{*}{\rightarrow}} q_\bullet' \in Q_f$$

   In particular, $r\tau \to^*_{\mathcal{A}} q'$. Recall that $\mathcal{A}$ is deterministic. Hence we can decompose this rewrite sequence as follows: $r\tau \to^*_{\mathcal{A}} r\sigma \to^*_{\mathcal{A}} q'$. We conclude by noting that $q \gg q'$ by the definition of $\gg$. ◻

Corollary 14. *The problem $\mathcal{R}(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A})$ is decidable.*

Proof. W.l.o.g. we may assume that $\mathcal{A}$ is deterministic. Using Proposition 7 we may also assume that $\mathcal{A}$ is trim. By Theorems 11 and 13 the problem reduces to whether there is some relation $\gg$ such that $(\mathcal{A}, \gg)$ is both state-compatible with $\mathcal{R}$ and state-coherent. But since there are only finitely many relations $\gg$ we can just test state-compatibility and state-coherence for each $\gg$.

Remark 15. As a consequence of Theorem 13, regular languages accepted by state-coherent automata that are state-compatible with a fixed TRS $\mathcal{R}$ are closed under intersection and union. This can also be shown directly by a product construction.

3.3 Deciding $\mathcal{R}(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A})$

In the remainder of this section we show that instead of testing all possible relations $\gg$, it suffices to construct a minimal one. We proceed as follows:

1. We assume that $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ is trim and deterministic. Note that given a non-deterministic automaton, we can compute an equivalent deterministic one in exponential time. Once we have a deterministic automaton, we can compute an equivalent trim one in polynomial time.
2. For each state $q \in Q$ and rule $l \to r \in \mathcal{R}$, check whether there is a state substitution σ such that $l\sigma \to_\Delta^* q$, but there is no q' with $r\sigma \to_\Delta^* q'$. If such a σ exists, then $\mathcal{L}(\mathcal{A})$ is not closed under rewriting by $\mathcal{R}$, and the procedure terminates.
3. In the following steps we will find the smallest relation $\gg$ that makes $(\mathcal{A}, \gg)$ both state-compatible with $\mathcal{R}$ and state-coherent, if such a relation exists.
4. For each pair of states $q, q' \in Q$ and rule $l \to r \in \mathcal{R}$, check whether there is a state substitution σ such that $l\sigma \to_\Delta^* q$ and $r\sigma \to_\Delta^* q'$. If so, *assert* $q \gg q'$. This ensures that $(\mathcal{A}, \gg)$ will be state-compatible with $\mathcal{R}$.
5. Whenever $q \gg q'$ is *asserted* for the first time for states q and q', we fail if q is final but q' is not, violating the state-coherence. Otherwise, we check Δ for transitions with q on the left-hand side. If $f(q_1, \ldots, q_i = q, \ldots, q_n) \to q_\bullet \in \Delta$, then we look for a transition with left-hand side $f(q_1, \ldots, q_i' = q', \ldots, q_n)$ in Δ. If no such transition exists, state-coherence fails, and the algorithm terminates. Otherwise, let $q_\bullet' \in Q$ be the corresponding right-hand side. We have $f(q_1, \ldots, q_i' = q', \ldots, q_n) \to q_\bullet' \in \Delta$. *Assert* that $q_\bullet \gg q_\bullet'$.

Note that step 5 is really a subroutine, and invokes itself recursively. Steps 2 and 4, which identify the applicable instances of the state-compatibility constraint, consist of a polynomial number of NP queries, and step 5 can be performed in polynomial time. The whole procedure is, therefore, in the Δ_2^{P} (or P^{NP}) complexity class for deterministic automata as input.

Remark 16. Using [3, Exercise 1.12.2], which shows that it is NP-hard to decide whether an instance of a term l is accepted by a tree automaton $\mathcal{A}$, we can

show that deciding whether the language accepted by a deterministic automaton is closed under rewriting by a given TRS is co-NP-hard. To wit, given a term l, a tree automaton $\mathcal{A}$, a fresh unary function $\star$ and a fresh constant $\diamond$, then $\star(\mathcal{L}(\mathcal{A})) = \{\star(x) \mid x \in \mathcal{L}(\mathcal{A})\}$ is closed under rewriting by $\star(l) \to \diamond$ if and only if no instance of l is accepted by $\mathcal{A}$.

4 Relation to Quasi-Deterministic Automata

We recall the definitions of compatibility and quasi-determinism from [10], and show that given a compatible, quasi-deterministic automaton, we can extract a state-compatible, deterministic automaton accepting the same language, while the opposite direction fails in the presence of collapsing rules.

Definition 17 (Definition 18 of [10]). *Let $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ be a tree automaton. For a left-hand side $l \in \mathrm{lhs}(\Delta)$ of a transition, we denote the set $\{q \mid l \to q \in \Delta\}$ of possible right-hand sides by $Q(l)$. We call $\mathcal{A}$* quasi-deterministic *if for every $l \in \mathrm{lhs}(\Delta)$ there exists a* designated state *$p \in Q(l)$ such that for all transitions $f(q_1, \ldots, q_n) \to q \in \Delta$ and $i \in \{1, \ldots, n\}$ with $q_i \in Q(l)$, the transition $f(q_1, \ldots, q_{i-1}, p, q_{i+1}, \ldots, q_n) \to q$ belongs to Δ. Moreover, we require that $p \in Q_f$ whenever $Q(l)$ contains a final state.*

For each $l \in \mathrm{lhs}(\Delta)$ we pick a state p_l satisfying the constraints of Definition 17. We denote the set of designated states by Q_d and the set $\{l \to p_l \mid l \in \mathrm{lhs}(\Delta)\}$ by Δ_d. The notion of compatibility used for quasi-deterministic tree automata is refined slightly from the standard one, Definition 4.

Definition 18 (Definition 23 of [10]). *Let $\mathcal{R}$ be a TRS and L a language. Let $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ be a quasi-deterministic tree automaton. We say that $\mathcal{A}$ is* compatible *with $\mathcal{R}$ and L if $L \subseteq \mathcal{L}(\mathcal{A})$ and for each rewrite rule $l \to r \in \mathcal{R}$ and state substitution $\sigma\colon \mathcal{V}\mathsf{ar}(l) \to Q_d$ such that $l\sigma \to^*_{\Delta_d} q$ it holds that $r\sigma \to^*_{\Delta} q$.*

Example 3 exhibits a quasi-deterministic, quasi-compatible automaton.

We will show that for each quasi-deterministic automaton that is compatible with a TRS $\mathcal{R}$, there is a deterministic, state-coherent automaton that is state-compatible with $\mathcal{R}$ and accepts the same language. To this end, we need the following key lemma, a slight generalization of [10, Lemma 20], which shows that a quasi-deterministic automaton $\mathcal{A}$ is almost deterministic: all but the last step in a reduction can be performed using the deterministic Δ_d transitions.

Lemma 19. *Let $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ be a quasi-deterministic automaton. If $t \to^+_{\Delta} q$ then $t \to^*_{\Delta_d} \cdot \to_{\Delta} q$ for all terms $t \in \mathcal{T}(\mathcal{F} \cup Q)$ and states $q \in Q$.*

Proof. Identical to the proof of [10, Lemma 20], except when t_i in $t = f(t_1, \ldots, t_n)$ is a state. In that case, we let $p_{l_i} = q_i = t_i$. □

Theorem 20. *Let $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ be a quasi-deterministic tree automaton that is compatible with $\mathcal{R}$. Then $\mathcal{A}' = (\mathcal{F}, Q_d, Q_f \cap Q_d, \Delta_d)$ makes $(\mathcal{A}', \gg)$ state-coherent and state-compatible with $\mathcal{R}$, where $q \gg q'$ if $q = q'$ or, for some left-hand side $l \in \mathrm{lhs}(\Delta)$, $q \in Q(l)$ and $q' = p_l$. Furthermore, $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$.*

Proof. Note that $\to_{\mathcal{A}} = \to_{\Delta}$ and $\to_{\mathcal{A}'} = \to_{\Delta_d}$.

1. (state-coherence) Assume that q is final in $\mathcal{A}'$, and $q \gg q'$. If $q = q'$ then q' is final, too. Otherwise, there is a left-hand side l such that $q \in Q(l)$ and $q' = p_l$ is the designated state of l. Since $Q(l)$ contains a final state (namely, q), q' must be final as well by Definition 17.
2. (state-coherence) Let $l = f(q_1, \ldots, q_i, \ldots, q_n)$ and $l' = f(q_1, \ldots, q_i', \ldots, q_n)$, where $q_i \gg q_i'$. Furthermore, let $l \to q \in \Delta_d$. If $q_i = q_i'$ then $l' \to q \in \Delta_d$ and $q \gg q$. Otherwise, there is a left-hand side $l^\bullet$ such that $q_i \in Q(l^\bullet)$ and $q_i' = p_{l^\bullet}$ is the designated state of $l^\bullet$. By Definition 17, there is a transition $l' \to q$ in Δ. Thus, l' is a left-hand side and $q \in Q(l')$. Furthermore, $l' \to p_{l'} \in \Delta_d$, and $q \gg p_{l'}$ follows.
3. (state-compatibility) Let σ be a state substitution and $l\sigma \to_{\Delta_d}^* q$. By compatibility, we have $r\sigma \to_{\Delta}^* q$. If r is a variable, we are done, noting that $q \gg q$. Otherwise, using Lemma 19, there is a left-hand side $l' \in \mathrm{lhs}(\mathcal{A})$ such that $r\sigma \to_{\Delta_d}^* l' \to_{\Delta} q$. Consequently, $r\sigma \to_{\Delta_d}^* \cdot \to_{\Delta_d} p_{l'}$, and since $q \in Q(l')$, we have $q \gg p_{l'}$.
4. (accepted language) $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$ is obvious. To show $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$, assume that $t \in \mathcal{L}(\mathcal{A})$, i.e., $t \to_{\Delta}^* q \in Q_f$. By Lemma 19, there is a left-hand side $l \in \mathrm{lhs}(\mathcal{A})$ such that $t \to_{\Delta_d}^* l \to_{\Delta} q$. As in the previous item we conclude that $t \to_{\Delta_d}^* p_l$, and $q \gg p_l$. The state p_l is final by state-coherence, so $t \in \mathcal{L}(\mathcal{A}')$ follows. □

In the opposite direction, we have a positive result for non-collapsing TRSs.

Theorem 21. *Let $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ be a deterministic automaton and the relation $\gg \subseteq Q \times Q$ be such that $(\mathcal{A}, \gg)$ is state-coherent and state-compatible with $\mathcal{R}$. Furthermore, assume that $\mathcal{R}$ contains no collapsing rules. Then the automaton $\mathcal{A}' = (\mathcal{F}, Q, Q_f, \Delta')$ with $\Delta' = \{l \to q' \mid l \to q \in \Delta, q \gg^= q'\}$ is a quasi-deterministic automaton with designated states $p_l = q$ for $l \to q \in \Delta$, such that $\mathcal{A}'$ is compatible with $\mathcal{R}$ and accepts the same language as $\mathcal{A}$.*

Proof. Verifying that the construction results in a quasi-deterministic automaton that is compatible with $\mathcal{R}$ is straight-forward. Note that applying Theorem 20 to $\mathcal{A}'$ results in some $(\mathcal{A}'', \gg'')$ with $\mathcal{L}(\mathcal{A}'') = \mathcal{L}(\mathcal{A}')$, where $\mathcal{A}''$ is $\mathcal{A}$ with states restricted to Q_d', the right-hand sides of Δ'. This restriction preserves the accepted language. Therefore, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. □

If $\mathcal{R}$ contains collapsing rules, quasi-deterministic, compatible automata may be weaker than state-coherent, state-compatible ones, as the following example demonstrates.

Example 22. Let $\mathcal{R} = \{\mathsf{f}(x, x) \to x\}$. The automaton $\mathcal{A}'$ over $\{\mathsf{f}, \mathsf{a}\}$ with states $1, 2$, both final, and transitions

$$\mathsf{a} \to 1 \qquad\qquad \mathsf{f}(1, 1) \to 2$$

accepts $L = \{\mathsf{f}(\mathsf{a}, \mathsf{a}), \mathsf{a}\}$. Furthermore, $(\mathcal{A}', \gg)$ is state-coherent and state-compatible with $\mathcal{R}$ if we let $2 \gg 1$.

Now assume that $\mathcal{A} = (\{\mathsf{f}, \mathsf{a}\}, Q, Q_f, \Delta)$ is a quasi-deterministic automaton and compatible with $\mathcal{R}$, and that $\mathsf{f}(\mathsf{a}, \mathsf{a}) \in \mathcal{L}(\mathcal{A})$. We will show that $\mathcal{A}$ accepts all terms over $\{\mathsf{f}, \mathsf{a}\}$. Note that since $\mathsf{f}(\mathsf{a}, \mathsf{a})$ is accepted, a must be a left-hand side of $\mathcal{A}$. Let q be the designated state of a. By Lemma 19, we have a run $\mathsf{f}(\mathsf{a}, \mathsf{a}) \to^*_{\Delta_d} \mathsf{f}(q, q) \to_\Delta q' \in Q_f$. Let $q^\bullet$ be the designated state of the left-hand side $\mathsf{f}(q, q)$. By quasi-determinism, $q^\bullet$ is a final state. Compatibility requires that $\mathsf{f}(q, q) \to_{\Delta_d} q^\bullet \ {}^*_\Delta\!\leftarrow q$, i.e., $q^\bullet = q$. So we have a final state q and two transitions $\mathsf{a} \to q$, $\mathsf{f}(q, q) \to q$, and $\mathcal{A}$ accepts all of $\mathcal{T}(\{\mathsf{f}, \mathsf{a}\})$.

Remark 23. In his thesis [9], Korp generalizes Definition 17 (cf. [9, Definition 3.10]) by incorporating an auxiliary relation $\succeq_{\phi_\mathcal{A}}$ that may be viewed as a precursor to our relation $\gg$. The modified definition permits smaller automata, which benefits implementations, but is more complicated than Definition 17. The modification also does not add expressive power. Indeed if $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ satisfies [9, Definition 3.10] using $\succeq_{\phi_\mathcal{A}}$, then taking $\Delta' = \{l \to q \mid l \in \mathrm{lhs}(\Delta), \phi_\mathcal{A}(l) \succeq q\}$, the automaton $\mathcal{A}' = (\mathcal{F}, Q, Q_f, \Delta')$ satisfies Definition 17, noting that $\phi_\mathcal{A}(l)$ is just another notation for the designated state p_l of l. Furthermore, $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$.

5 Formalization

We have formalized all results from Section 3 as part of IsaFoR, our *Isa*belle *Fo*rmalization of *R*ewriting, in combination with executable algorithms which check state-compatibility and state-coherence. These are used in CeTA [14], a certifier for several properties related to term rewriting.

Here, `Tree_Automata.thy` starts with basic definitions on tree automata where there are two major differences to this paper: the formalization allows ε-transitions, and the set of *re*achable *s*tates $t \to^*_\Delta q$ is formalized directly as a function *ta_res* mapping terms to sets of states. Using a function instead of a relation has both positive and negative effects. For example, it eases proofs which are naturally performed by induction on terms, since in $f(t_1, \ldots, t_n)$ one does not have to reduce all arguments t_1 to t_n sequentially in a relation, but this is done in one step in *ta_res*. On the other hand, one cannot trace derivations $t \to^*_\Delta q$ explicitly as there is no notion of derivation. Hence, some obvious results have to be proven explicitly by induction, e.g., that removing transitions results in a smaller accepted language.

The file continues with proofs of Proposition 7 (`obtain_trimmed_ta`), Theorems 11 and 13 (`state_compatible_lang` and `ta_trim_det_closed`), and Corollary 14 (`closed_iff_compatible_and_coherent`), where the corollary states only that $\mathcal{L}(\mathcal{A})$ is closed under $\mathcal{R}$ iff for the determinized and trimmed automaton there exists a suitable relation $\gg$. Instead of formally proving decidability by an algorithm which enumerates all possible relations, we directly formalized the algorithm of Section 3.3 to compute the least such relation. Here, we described the algorithm on an abstract level via some inference system, and its soundness and completeness manifests in theorem `decide_coherent_compatible`. It is later on refined to a fully executable one.

In addition to the decision procedure, we also provide Theorem 11 to demonstrate closure under rewriting when $\gg$ is supplied. The advantage of the latter is its improved runtime and its broader applicability: one does not have to iteratively construct the relation, and for left-linear TRSs, also non-deterministic automata with ε-transitions are supported, cf. `state_compatible_lang`. Here, for checking state-compatibility, we use a tree automaton matching algorithm (*ta_match*), that restricts the set of state substitutions σ that have to be considered for compatibility w.r.t. Definition 8.

Whereas `Tree_Automata.thy` formalizes most algorithms on an abstract level, in `Tree_Automata_Impl.thy` we refined those to fully executable ones. In fact, for some algorithms we just relied on the automatic refinement provided in [12] which turns set operations into operations on trees. However, for some algorithms like the matching algorithm we performed manual refinement to increase the efficiency. For example, we group the transitions of an automaton by their root symbols and store these groups in ordered trees using Isabelle's collection framework [11]. Moreover, for each $f(q_1, \ldots, q_n) \to q$, we precompute the closure of q under ε-transitions. This speeds up the computation of *ta_res* while checking state-compatibility. In the end, we provide an executable algorithm which for given $\mathcal{A}$ and $\mathcal{R}$ checks whether $\mathcal{A}$ is closed under $\mathcal{R}$, cf. `tree_aut_trs_closed`.

We have extended the termination tool T$_{\text{T}}$T$_2$ [8] and the confluence tool CSI [15] to produce state-coherent, state-compatible automata. Since both tools use quasi-deterministic automata in their completion process, we applied the construction of Theorem 20 as a post-processing step, resulting in a state-coherent, state-compatible automaton. CeTA can then be used to certify this output. Whereas for non-confluence proofs the input can be arbitrary, for match-bounds we currently require left-linearity. The reason is that without left-linearity, the match-bounds technique requires further conditions besides closure under rewriting, which have not been formalized yet and which remain as future work.

All tools and the formalization are available at `http://cl-informatik.uibk.ac.at/research/software/` (CeTA + IsaFoR version 2.12, T$_{\text{T}}$T$_2$ version 1.14, CSI version 0.4.)

6 Conclusion

We have introduced the class of deterministic, state-coherent automata that are state-compatible with a TRS $\mathcal{R}$. We have shown that these automata capture precisely those regular tree languages that are closed under rewriting by $\mathcal{R}$, leading to a decision procedure for checking whether a regular language is closed under rewriting. Their simple definition allowed us to formalize most of our results on state-coherent, state-compatible automata.

Even though state-coherent, state-compatible tree automata are strictly more general, we still rely on quasi-deterministic tree automata for the actual completion process in the CSI and T$_{\text{T}}$T$_2$ tools. Thus, they cannot exploit the full power of state-coherent and state-compatible tree automata, and will fail when analyzing TRSs like Example 22. As future work, we plan to investigate whether

working directly on state-coherent, state-compatible automata can improve tree automata completion.

Acknowledgments. We would like to thank Aart Middeldorp for fruitful discussions on the topic of tree automata and helpful feedback. We are also grateful to the anonymous reviewers for their constructive feedback.

References

1. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
2. Boyer, B., Genet, T., Jensen, T.P.: Certifying a tree automata completion checker. In: Proc. 4th International Joint Conference on Automated Reasoning. Lecture Notes in Computer Science, vol. 5195, pp. 523–538 (2008)
3. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Löding, C., Tison, S., Tommasi, M.: Tree Automata Techniques and Applications (2007), available at `http://tata.gforge.inria.fr`
4. Feuillade, G., Genet, T., Tong, V.V.T.: Reachability analysis over term rewriting systems. Journal of Automated Reasoning 33, 341–383 (2004)
5. Genet, T.: Decidable approximations of sets of descendants and sets of normal forms. In: Proc. 9th International Conference on Rewriting Techniques and Applications. Lecture Notes in Computer Science, vol. 1379, pp. 151–165 (1998)
6. Genet, T., Tang-Talpin, Y.M., Tong, V.V.T.: Verification of copy-protection cryptographic protocol using approximations of term rewriting systems. In: Proc. WITS'03 (Workshop on Issues in the Theory of Security) (2003)
7. Geser, A., Hofbauer, D., Waldmann, J., Zantema, H.: On tree automata that certify termination of left-linear term rewriting systems. Information and Computation 205(4), 512–534 (2007)
8. Hirokawa, N., Middeldorp, A.: Tyrolean termination tool. In: Proc. 16th International Conference on Rewriting Techniques and Applications. Lecture Notes in Computer Science, vol. 3467, pp. 175–184 (2005)
9. Korp, M.: Termination Analysis by Tree Automata Completion. Ph.D. thesis, University of Innsbruck (2010)
10. Korp, M., Middeldorp, A.: Match-bounds revisited. Information and Computation 207(11), 1259–1283 (2009)
11. Lammich, P., Lochbihler, A.: The Isabelle collections framework. In: Proc. 1st International Conference on Interactive Theorem Proving. Lecture Notes in Computer Science, vol. 6172, pp. 339–354 (2010)
12. Lochbihler, A.: Light-weight containers for Isabelle: Efficient, extensible, nestable. In: Proc. 4th International Conference on Interactive Theorem Proving. Lecture Notes in Computer Science, vol. 7998, pp. 116–132 (2013)
13. Nipkow, T., Paulson, L., Wenzel, M.: Isabelle/HOL – A Proof Assistant for Higher-Order Logic, Lecture Notes in Computer Science, vol. 2283. Springer (2002)
14. Thiemann, R., Sternagel, C.: Certification of termination proofs using CeTA. In: Proc. 22nd International Conference on Theorem Proving in Higher Order Logics. Lecture Notes in Computer Science, vol. 5674, pp. 452–468 (2009)
15. Zankl, H., Felgenhauer, B., Middeldorp, A.: CSI – A confluence tool. In: Proc. 23rd International Conference on Automated Deduction. Lecture Notes in Artificial Intelligence, vol. 6803, pp. 499–505 (2011)