Completeness of Tree Automata Completion
Thomas Genet

To cite this version:

HAL Id: hal-01778407
https://hal.science/hal-01778407
Submitted on 25 Apr 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Completeness of Tree Automata Completion

Thomas Genet
Univ Rennes/Inria/IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France
genet@irisa.fr

Abstract
We consider rewriting of a regular language with a left-linear term rewriting system. We show a completeness theorem on equational tree automata completion stating that, if there exists a regular over-approximation of the set of reachable terms, then equational completion can compute it (or safely under-approximate it). A nice corollary of this theorem is that, if the set of reachable terms is regular, then equational completion can also compute it. This was known to be true for some term rewriting system classes preserving regularity, but was still an open question in the general case. The proof is not constructive because it depends on the regularity of the set of reachable terms, which is undecidable. To carry out those proofs we generalize and improve two results of completion: the Termination and the Upper-Bound theorems. Those theoretical results provide an algorithmic way to safely explore regular approximations with completion. This has been implemented in Timbuk and used to verify safety properties, automatically and efficiently, on first-order and higher-order functional programs.

2012 ACM Subject Classification
I.2.3 Deduction and Theorem Proving, F.4.2 Grammars and Other Rewriting Systems

Keywords and phrases term rewriting systems, regularity preservation, over-approximation, completeness, tree automata, tree automata completion

Digital Object Identifier 10.4230/LIPIcs.FSCD.2018.15

1 Introduction

Given a term rewriting system (TRS for short) \( R \) and a tree automaton \( A \) recognizing a regular tree language \( \mathcal{L}(A) \), the set of reachable terms is \( R^*(\mathcal{L}(A)) = \{ t \mid s \in \mathcal{L}(A) \text{ and } s \rightarrow_R^* t \} \). In this paper, we show that the *equational tree automata completion algorithm* [15] is complete w.r.t. regular approximations. If \( R \) is left-linear and there exists a regular language \( \mathcal{L} \) over-approximating \( R^*(\mathcal{L}(A)) \), i.e., \( R^*(\mathcal{L}(A)) \subseteq \mathcal{L} \) then completion can build a tree automaton \( A^* \) such that \( R^*(\mathcal{L}(A)) \subseteq \mathcal{L}(A^*) \subseteq \mathcal{L} \). We also show that completion is complete w.r.t. TRSs preserving regularity, i.e., if \( \mathcal{L} = R^*(\mathcal{L}(A)) \) then completion can build a tree automaton \( A^* \) such that \( R^*(\mathcal{L}(A)) = \mathcal{L}(A^*) = \mathcal{L} \). On the one hand, automata built by completion-like algorithms are known to recognize exactly the set of reachable terms, for some restricted classes of TRSs [17, 24, 8, 10]. On the other hand, automata completion is able to build over-approximations for any left-linear TRS [9, 23, 15], and even for non-left-linear TRSs [3]. Such approximations are used for program verification [5, 4, 10, 14] as well as to automate termination proofs [16, 20]. To define approximations, completion uses an additional set of equations \( E \) and builds a tree automaton \( A^*_{R,E} \) such that \( \mathcal{L}(A^*_{R,E}) \supseteq R^*(\mathcal{L}(A)) \). Starting from \( R \), \( A \), and \( E \) Timbuk[12] is an automatic tool to build \( A^*_{R,E} \). Until now it was an open question whether completion can build any regular over-approximation or compute the set of reachable terms if this set is regular. The *first contribution* of this paper is to answer these two questions in the positive, for general left-linear TRSs. The proofs are not constructive but, the *second contribution* is to provide an efficient method to explore regular approximations for TRSs encoding functional programs.
For the approximated case, the proof of completeness is organized as follows. If there exists a regular over-approximation \( \mathcal{L} \) such that \( \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L} \), we know that there exists a tree automaton \( \mathcal{B} \) such that \( \mathcal{L}(\mathcal{B}) = \mathcal{L} \). From \( \mathcal{B} \), using the Myhill-Nerode theorem, we can infer a set of equations \( E \) such that the set of \( E \)-equivalence classes \( \mathcal{T}(\mathcal{F})/_{\mathcal{E}} \) is finite. Then we prove the following theorems:

(a) If \( \mathcal{T}(\mathcal{F})/_{\mathcal{E}} \) is finite, then it is possible to build from \( E \) a set of equations \( E' \), equivalent to \( E \), such that completion of any automaton \( \mathcal{A} \) by any TRS \( \mathcal{R} \) with \( E' \) always terminates. This generalizes the termination theorem of [10];

(b) If \( \mathcal{T}(\mathcal{F})/_{\mathcal{E}} \) is finite, then it is possible to build from \( E \) and \( \mathcal{A} \) a tree automaton \( \mathcal{A} \) recognizing the same language as \( \mathcal{A} \) such that the completed automaton \( \mathcal{A}_E^* \mathcal{R} \mathcal{E} \) has the following precision property: \( \mathcal{L}(\mathcal{A}_E^* \mathcal{R} \mathcal{E}) \subseteq \mathcal{R}_E^*(\mathcal{L}(\mathcal{A})) \), where \( \mathcal{R}_E^*(\mathcal{L}(\mathcal{A})) \) is the set of reachable terms by rewriting modulo \( E \). It generalizes the Upper Bound theorem of [15].

(c) Then, we show that \( \mathcal{R}_E^*(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{B}) \), and we get the main completeness theorem: \( \mathcal{L}(\mathcal{A}_E^* \mathcal{R} \mathcal{E}) \subseteq \mathcal{R}_E^*(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{B}) \).

Besides, we know from [15] that \( \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A}_E^* \mathcal{R} \mathcal{E}) \). Thus, when using the set of equations defined from \( \mathcal{B} \) to run completion, (c) implies that we can only get an over-approximation of \( \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \) equivalent or better than \( \mathcal{L} = \mathcal{L}(\mathcal{B}) \). This result has a practical impact for program verification. In particular, for TRSs encoding functional programs, the search space of sets of equations \( E \) can be constrained for enumeration to be possible. This has been implemented in the Timbuk [12] tool. Our experiments show that this makes completion automatic enough to carry out safety proofs on first-order and higher-order functional programs. We also get a corollary of (c) when \( \mathcal{L} \) is not an approximation:

(d) If \( \mathcal{L} = \mathcal{L}(\mathcal{B}) = \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \), we can use \( \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A}_E^* \mathcal{R} \mathcal{E}) \) to close-up the \( \subseteq \)-chain and get that \( \mathcal{L}(\mathcal{A}_E^* \mathcal{R} \mathcal{E}) = \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \). Thus if \( \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \) is regular, there exists a set of equations \( E \) s.t. \( \mathcal{L}(\mathcal{A}_E^* \mathcal{R} \mathcal{E}) = \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \)

Section 2 defines some basic notions in term rewriting and tree automata and Section 3 recalls the tree automata completion algorithm and the related theorems. Section 4 recalls the Myhill-Nerode theorem for trees and defines the functions to transform a set of equations into a tree automaton and vice versa. Section 5 proves Result (a) and Section 6 shows Result (b). Section 7 assembles (a) and (b) to prove results (c) and (d) using the proof sketched above. Section 8 shows how to take advantage of those results to program verification and presents some experiments. Finally, Section 9 concludes.

## 2 Preliminaries

In this section we introduce some definitions and concepts that will be used throughout the rest of the paper (see also [2, 6]). Let \( \mathcal{F} \) be a finite set of symbols, each associated with an arity function. For brevity, we write \( f : n \) if \( f \) is a symbol of arity \( n \) and \( \mathcal{F}^n = \{ f \in \mathcal{F} \mid f : n \} \). Let \( \mathcal{X} \) be a countable set of variables, \( \mathcal{T}(\mathcal{F}, \mathcal{X}) \) denotes the set of terms and \( \mathcal{T}(\mathcal{F}) \) denotes the set of \textit{ground terms} (terms without variables). The set of variables of a term \( t \) is denoted by \( \mathcal{V}(t) \). A \textit{substitution} is a function \( \sigma \) from \( \mathcal{X} \) into \( \mathcal{T}(\mathcal{F}, \mathcal{X}) \), which can be uniquely extended to an endomorphism of \( \mathcal{T}(\mathcal{F}, \mathcal{X}) \). A \textit{position} \( p \) in a term \( t \) is a finite word over \( \mathbb{N} \), the set of natural numbers. The empty sequence \( \lambda \) denotes the top-most position. The set \( \mathcal{P}(t) \) of positions of a term \( t \) is inductively defined by \( \mathcal{P}(t) = \{ \lambda \} \) if \( t \in \mathcal{X} \) or \( t \) is a constant and \( \mathcal{P}(f(t_1, \ldots, t_n)) = \{ \lambda \} \cup \{ i, p \mid 1 \leq i \leq n \text{ and } p \in \mathcal{P}(t_i) \} \) otherwise. If \( p \in \mathcal{P}(t) \), then \( t(p) \) denotes the symbol at position \( p \) in \( t \), \( t|_p \) denotes the
subterm of $t$ at position $p$, and $t[s]_p$ denotes the term obtained by replacing the subterm $t|_p$ at position $p$ by the term $s$. A ground context $C[]$ is a term in $\mathcal{T}(\mathcal{F} \cup \{\Box\})$ containing exactly one occurrence of the symbol $\Box$. If $t \in \mathcal{T}(\mathcal{F})$ then $C[t]$ denotes the term obtained by the replacement of $\Box$ by $t$ in $C[]$. A context is empty if it is equal to $\Box$.

A term rewriting system (TRS) $\mathcal{R}$ is a set of rewrite rules $l \rightarrow r$, where $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $l \not\in \mathcal{X}$, and $\forall \text{ar}(l) \supseteq \forall \text{ar}(r)$. A rewrite rule $l \rightarrow r$ is left-linear if each variable occurs only once in $l$. A TRS $\mathcal{R}$ is left-linear if every rewrite rule $l \rightarrow r$ of $\mathcal{R}$ is left-linear. The TRS $\mathcal{R}$ induces a rewriting relation $\rightarrow_{\mathcal{R}}$ on terms as follows. Let $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and $l \rightarrow r \in \mathcal{R}$, $s \rightarrow_{\mathcal{R}} t$ denotes that there exists a position $p \in \text{Pos}(s)$ and a substitution $\sigma$ such that $s|_p = lr\sigma$ and $t = s[r\sigma]_p$. The set of ground terms irreducible by a TRS $\mathcal{R}$ is denoted by $\text{IRR}(\mathcal{R})$. A set $\mathcal{L} \subseteq \mathcal{T}(\mathcal{F})$ is $\mathcal{R}$-closed if for all $s \in \mathcal{L}$ and $s \rightarrow_{\mathcal{R}} t$ then $t \in \mathcal{L}$. The reflexive transitive closure of $\rightarrow_{\mathcal{R}}$ is denoted by $\rightarrow_{\mathcal{R}}^*$, and $s \rightarrow_{\mathcal{R}}^* t$ denotes that $s \rightarrow_{\mathcal{R}}^* t$ and $t$ is irreducible by $\mathcal{R}$. The set of $\mathcal{R}$-descendants of a set of ground terms $I$ is defined as $\mathcal{R}^*(I) = \{t \in \mathcal{T}(\mathcal{F}) \mid \exists s \in I \text{ s.t. } s \rightarrow_{\mathcal{R}}^* t\}$, i.e., the smallest $\mathcal{R}$-closed set containing $I$.

Let $E$ be a set of equations $l = r$, where $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. The relation $\approx_E$ is the smallest congruence such that for all equations $l = r$ of $E$ and for all substitutions $\sigma$ we have $l\sigma = r\sigma$. The set of equivalence classes defined by $\approx_E$ on $\mathcal{T}(\mathcal{F})$ is denoted by $\mathcal{T}(\mathcal{F})/\approx_E$.

Given a TRS $\mathcal{R}$ and a set of equations $E$, a term $s \in \mathcal{T}(\mathcal{F})$ is rewritten modulo $E$ into $t \in \mathcal{T}(\mathcal{F})$, denoted $s \rightarrow_{\mathcal{R}/E} t$, if there exists an $s' \in \mathcal{T}(\mathcal{F})$ and a $t' \in \mathcal{T}(\mathcal{F})$ such that $s \approx_E s' \rightarrow_{\mathcal{R}} t' \approx_E t$. The reflexive transitive closure $\rightarrow_{\mathcal{R}/E}^*$ of $\rightarrow_{\mathcal{R}/E}$ is defined as usual except that reflexivity is extended to terms equal modulo $E$, i.e., for all $s, t \in \mathcal{T}(\mathcal{F})$, if $s \approx_E t$ then $s \rightarrow_{\mathcal{R}/E}^* t$. The set of $\mathcal{R}$-descendants modulo $E$ of a set of ground terms $I$ is defined as $\mathcal{R}^*_E(I) = \{t \in \mathcal{T}(\mathcal{F}) \mid \exists s \in I \text{ s.t. } s \rightarrow_{\mathcal{R}/E}^* t\}$.

Let $Q$ be a countably infinite set of symbols with arity 0, called states, such that $Q \cap \mathcal{F} = \emptyset$. Terms in $\mathcal{T}(\mathcal{F} \cup Q)$ are called configurations. A transition is a rewrite rule $c \rightarrow q$, where $c$ is a configuration and $q$ is a state. A transition is normalized when $c = f(q_1, \ldots, q_n), f \in \mathcal{F}$ is of arity $n$, and $q_1, \ldots, q_n \in Q$. An $\epsilon$-transition is a transition of the form $q \rightarrow q'$ where $q$ and $q'$ are states. A bottom-up non-deterministic finite tree automaton (tree automaton for short) over the alphabet $\mathcal{F}$ is a tuple $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$, where $Q_f \subseteq Q$ is the set of final states, $\Delta$ is a finite set of normalized transitions and $\epsilon$-transitions. An automaton is epsilon-free if it is free of $\epsilon$-transitions. The transitive and reflexive rewriting relation on $\mathcal{T}(\mathcal{F} \cup Q)$ induced by the set of transitions $\Delta$ (resp. all transitions except $\epsilon$-transitions) is denoted by $\rightarrow_{\Delta}^*$ (resp. $\rightarrow_{\Delta}^{\epsilon}$). When $\Delta$ is attached to a tree automaton $\mathcal{A}$ we also denote those two relations by $\rightarrow_{\Delta}^*$ and $\rightarrow_{\Delta}^{\epsilon}$, respectively. A tree automaton $\mathcal{A}$ is complete if for all $s \in \mathcal{T}(\mathcal{F})$ there exists a state $q$ of $\mathcal{A}$ such that $s \rightarrow_{\Delta}^* q$. The language recognized by $\mathcal{A}$ in a state $q$ is defined by $\mathcal{L}(\mathcal{A}, q) = \{t \in \mathcal{T}(\mathcal{F}) \mid t \rightarrow_{\Delta}^* q\}$. We define $\mathcal{L}(\mathcal{A}) = \bigcup_{q \in Q_f} \mathcal{L}(\mathcal{A}, q)$. A state $q$ of an automaton $\mathcal{A}$ is reachable if $\mathcal{L}(\mathcal{A}, q) \neq \emptyset$. An automaton is reduced if all its states are reachable. An automaton $\mathcal{A}$ is $\epsilon$-reduced if for all states $q$ of $\mathcal{A}$ there exists a ground term $t \in \mathcal{T}(\mathcal{F})$ such that $t \rightarrow_{\Delta}^{\epsilon} q$. An automaton $\mathcal{A}$ is deterministic if for all ground terms $s \in \mathcal{T}(\mathcal{F})$ and all states $q, q'$ of $\mathcal{A}$, if $s \rightarrow_{\Delta}^* q$ and $s \rightarrow_{\Delta}^* q'$ then $q = q'$. An automaton $\mathcal{A}$ is $\mathcal{R}$-closed if for all terms $s, t$ and all states $q \in Q$, $s \rightarrow_{\Delta}^* q$ and $s \rightarrow_{\mathcal{R}}^* t$ implies $t \rightarrow_{\Delta}^* q$.

## 3 Equational Tree Automata Completion

From a tree automaton $\mathcal{A}_0 = (\mathcal{F}, Q, Q_f, \Delta_0)$ and a left-linear TRS $\mathcal{R}$, the completion algorithm computes an automaton $\mathcal{A}^*$ such that $\mathcal{L}(\mathcal{A}^*) = \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$ or $\mathcal{L}(\mathcal{A}^*) \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$. 

---

**FSCD 2018**
3.1 Completion General Principles

From $\mathcal{A}_R^0 = \mathcal{A}_0$, tree automata completion successively computes tree automata $\mathcal{A}_R^1$, $\mathcal{A}_R^2$, \ldots such that for all $i \geq 0$: $L(\mathcal{A}_R^i) \subseteq L(\mathcal{A}_R^{i+1})$ and if $s \in L(\mathcal{A}_R^i)$, and $s \rightarrow_R t$ then $t \in L(\mathcal{A}_R^{i+1})$. For $k \in \mathbb{N}$, if $L(\mathcal{A}_R^k) = L(\mathcal{A}_R^{k+1})$ then $\mathcal{A}_R^k$ is a fixpoint and we denote it by $\mathcal{A}_R^*$. To construct $\mathcal{A}_R^{i+1}$ from $\mathcal{A}_R^i$, we perform a completion step (denoted by $\mathcal{C}_R$) which consists in finding critical pairs between $\rightarrow_R$ and $\rightarrow_{A_R}$. For a substitution $\sigma : \mathcal{X} \mapsto \mathcal{Q}$ and a rule $l \rightarrow r \in \mathcal{R}$, a critical pair is an instance $lr \sigma$ of $l$ such that there exists a state $q \in \mathcal{Q}$ satisfying $lr \sigma \rightarrow_{A_R} q$ and $r \sigma \not\rightarrow_{A_R} q$. For $r \sigma$ to be recognized by the same state and thus model the rewriting of $lr \sigma$ into $rr \sigma$, it is enough to add the necessary transitions to $\mathcal{A}_R^i$ in order to obtain $\mathcal{A}_R^{i+1}$ such that $rr \sigma \rightarrow_{\mathcal{A}_R^{i+1}} q$. The result of the completion step $\mathcal{C}_R(\mathcal{A}_R^i)$ is thus $\mathcal{A}_R^{i+1}$. In [24, 15], critical pairs are joined as in Figure 1.

From an algorithmic point of view, there remain two problems to solve: find all the critical pairs (1 → r, σ, q) and find the transitions to add to $\mathcal{A}_R^i$ to have $rr \sigma \rightarrow_{\mathcal{A}_R^{i+1}} q$. The first problem, called matching, can be efficiently solved using a specific algorithm [8]. The second problem is solved using a normalization algorithm [10]. To have $rr \sigma \rightarrow_{\mathcal{A}_R^{i+1}} q'$ we need a transition of the form $rr \sigma \rightarrow q'$ in $\mathcal{A}_R^{i+1}$. However, this transition may not be normalized. In this case, it is necessary to introduce new states and new transitions. For instance, to normalize a transition $f(g(a), h(q_1)) \rightarrow q'$ w.r.t. a tree automaton $\mathcal{A}_R^i$ with transitions $a \rightarrow q_1, b \rightarrow q_1$, $g(q_1) \rightarrow q_1$, we first rewrite $f(g(a), h(q_1))$ with transitions of $\mathcal{A}_R^i$ as far as possible. We obtain $f(q_1, h(q_1))$. Then we introduce the new state $q_2$ and the new transition $h(q_1) \rightarrow q_2$ to recognize the term $h(q_1)$. The new transitions to add to $\mathcal{A}_R^i$ are thus: $h(q_1) \rightarrow q_2, f(q_1, q_2) \rightarrow q'$, and $q' \rightarrow q$.

3.2 Simplification of Tree Automata by Equations

Since completion creates new transitions and new states to join critical pairs, it may diverge. Divergence is avoided by simplifying the tree automaton with a set of equations $E$. This operation permits the over-approximation of languages that cannot be recognized exactly using tree automata completion, e.g., non-regular languages. Simplification consists in finding $E$-equivalent terms recognized in $\mathcal{A}$ by different states and then by merging those states.

**Definition 1** (Simplification relation). Let $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, Q_f, \Delta \rangle$ be a tree automaton and $E$ be a set of equations. For $s = t \in E$, $\sigma : \mathcal{X} \mapsto \mathcal{Q}$, $q_a, q_b \in \mathcal{Q}$ such that $ss \rightarrow_{\mathcal{A}} q_a, tt \rightarrow_{\mathcal{A}} q_b$ (See Figure 2) and $q_a \not\equiv q_b$ then $\mathcal{A}$ is simplified into $\mathcal{A}'$, denoted by $\mathcal{A} \sim_E \mathcal{A}'$, where $\mathcal{A}'$ is $\mathcal{A}$ where $q_b$ is replaced by $q_a$ in $\mathcal{Q}$, $Q_f$ and $\Delta$.

**Example 2.** Let $E = \{ s(s(x)) = s(x) \}$ and $\mathcal{A}$ be the tree automaton with $Q_f = \{ q_2 \}$ and set of transitions $\Delta = \{ a \rightarrow q_0, s(q_0) \rightarrow q_1, s(q_1) \rightarrow q_2 \}$. Hence $L(\mathcal{A}) = \{ s(s(a)) \}$. We can perform a simplification step using the equation $s(s(x)) = s(x)$ because we found a substitution $\sigma = \{ x \mapsto q_0 \}$ such that $s(s(q_0)) \rightarrow_{\mathcal{A}} q_2, s(q_0) \rightarrow q_1$ (see Figure 3). Hence, $\mathcal{A} \sim_E \mathcal{A}'$ where $\mathcal{A}'$ is $\mathcal{A}$ where $q_2$ is replaced by $q_1$ i.e., $\mathcal{A}'$ is the automaton with $Q_f' = \{ q_1 \}$, $\Delta = \{ a \rightarrow q_0, s(q_0) \rightarrow q_1, s(q_1) \rightarrow q_1 \}$. Note that $L(\mathcal{A}') = \{ s'(s(a)) \}$.
The simplification relation $\sim_E$ is terminating and confluent (modulo state renaming) [15].
In the following, by $S_E(A)$ we denote the unique automaton (modulo renaming) $A'$ such
that $A \sim_E A'$ and $A'$ is irreducible (it cannot be simplified further).

### 3.3 The full Completion Algorithm

**Definition 3 (Automaton completion).** Let $A$ be a tree automaton, $R$ a left-linear TRS
and $E$ a set of equations.

- $A_{R,E}^0 = A$,
- $A_{R,E}^{n+1} = S_E(C_R(A_{R,E}^n))$, for $n \geq 0$ where $C_R(A_{R,E}^n)$ is the tree automaton such that all
critical pairs of $A_{R,E}^n$ are joined.

If there exists $k \in \mathbb{N}$ such that $A_{R,E}^k = A_{R,E}^{k+1}$, then we write $A_{R,E}^\infty$ for $A_{R,E}^k$.

**Example 4.** Let $R = \{ f(x,y) \rightarrow f(s(x),s(y)) \}$, $E = \{ s(s(x)) = s(x) \}$ and $A^0$ be
the tree automaton with set of transitions $\Delta = \{ f(q_0,q_0) \rightarrow q_0, a \rightarrow q_0, b \rightarrow q_0 \}$, i.e.,
$L(A^0) = \{ f(a,b) \}$. The completion ends after two completion steps on $A_{R,E}^2$ which is a
fixpoint $A_{R,E}^\infty$. Completion steps are summed up in the following table. To simplify the
presentation, we do not repeat the common transitions: $A_{R,E}^0 = A_{R,E}^{i-1}$.

<table>
<thead>
<tr>
<th>$A^0$</th>
<th>$C_R(A^0)$</th>
<th>$A_{R,E}^2$</th>
<th>$C_R(A_{R,E}^2)$</th>
<th>$A_{R,E}^\infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(q_0,q_0) \rightarrow q_0$</td>
<td>$f(q_1,q_2) \rightarrow q_3$</td>
<td>$f(q_1,q_2) \rightarrow q_3$</td>
<td>$f(q_4,q_5) \rightarrow q_6$</td>
<td>$f(q_1,q_2) \rightarrow q_6$</td>
</tr>
<tr>
<td>$a \rightarrow q_0$</td>
<td>$s(q_0) \rightarrow q_1$</td>
<td>$s(q_0) \rightarrow q_1$</td>
<td>$s(q_4) \rightarrow q_4$</td>
<td>$s(q_4) \rightarrow q_4$</td>
</tr>
<tr>
<td>$b \rightarrow q_0$</td>
<td>$s(q_0) \rightarrow q_2$</td>
<td>$s(q_0) \rightarrow q_2$</td>
<td>$s(q_2) \rightarrow q_5$</td>
<td>$s(q_2) \rightarrow q_5$</td>
</tr>
<tr>
<td>$q_3 \rightarrow q_0$</td>
<td>$q_3 \rightarrow q_0$</td>
<td>$q_3 \rightarrow q_0$</td>
<td>$q_6 \rightarrow q_1$</td>
<td>$q_6 \rightarrow q_1$</td>
</tr>
</tbody>
</table>

On $A^0$, there is one critical pair $f(q_0,q_0) \rightarrow A^0 q_0$ and $f(q_0,q_0) \rightarrow R s(q_0)$.
The automaton $C_R(A^0)$ contains all the transitions of $A^0$ with the new transitions (and the
new states) necessary to join the critical pair, i.e., to have $f(s(q_0), s(q_0)) \rightarrow C_R(A^0) q_0$.
The automaton $A_{R,E}^1$ is exactly $C_R(A^0)$ because simplification by $E$ does not apply. Then,
$C_R(A_{R,E}^1)$ contains all the transitions of $A_{R,E}^1$ and $A^0$ plus those obtained by the resolution
of the critical pair $f(q_1,q_2) \rightarrow A_{R,E}^1 q_3$ and $f(q_1,q_2) \rightarrow R s(q_1,q_2)$. On $C_R(A_{R,E}^1)$
simplification using the equation $s(s(x)) = s(x)$ can be applied on the following instances:
$s(s(q_0)) = s(q_0)$ and $s(s(q_0)) = q_0$. Since $s(s(q_0)) \rightarrow C_R(A_{R,E}^1) q_4$ and $s(q_0) \rightarrow C_R(A_{R,E}^1) q_1$,
simplification merges $q_4$ with $q_1$. Similarly, simplification on $s(s(q_0)) = q_0$ merges $q_4$ with $q_2$.
Thus, $A_{R,E}^2 = C_R(A_{R,E}^1)$ where $q_4$ is replaced by $q_1$ and $q_0$ is replaced by $q_2$.
This automaton is a fixed point because it has no other critical pairs (they are all joined).

### 3.4 Lower Bound, Upper Bound and Termination of Completion

**Theorem 5 (Lower Bound [15]).** Let $R$ be a left-linear TRS, $A$ be a tree automaton and
$E$ be a set of equations. If completion terminates on $A_{R,E}^\infty$ then $L(A_{R,E}^\infty) \supseteq R^*(L(A))$.

To state the upper bound theorem, we need the notion of $R/E$-coherence we now define.

**Definition 6 (Coherent automaton).** Let $A = \langle F, Q, \Lambda, \Delta \rangle$ be a tree automaton, $R$ a
TRS and $E$ a set of equations. The automaton $A$ is said to be $R/E$-coherent if $\forall q \in Q : \exists s \in T(F) :$

\[
\forall t \in T(F) : (t \rightarrow_A^* q \Rightarrow s = E t) \land (t \rightarrow_A^* q \Rightarrow s \rightarrow_{R/E} t).
\]
Completeness of Tree Automata Completion

Here is the intuition behind $\mathcal{R}/E$-coherence. An $\mathcal{R}/E$-coherent automaton is $\kappa$-reduced, its $\epsilon$-transitions represent rewriting steps and normalized (\(\kappa\)-transitions) transitions recognize $E$-equivalence classes. More precisely, in an $\mathcal{R}/E$-coherent tree automaton, if two terms $s, t$ are recognized in the same state $q$ using only normalized transitions then they belong to the same $E$-equivalence class. Otherwise, if at least one $\epsilon$-transition is necessary to recognize, say, $t$ in $q$ then at least one step of rewriting with $\mathcal{R}$ was necessary to obtain $t$ from $s$.

**Example 7.** Let $\mathcal{R} = \{a \to b\}$, $E = \{c = d\}$ and $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ with $\Delta = \{a \to q_0, b \to q_1, c \to q_2, d \to q_2, q_1 \to q_0\}$. The automaton $\mathcal{A}$ is $\mathcal{R}/E$-coherent because it is $\kappa$-reduced and the state $q_2$ recognizes with $\rightarrow_A^1$ two terms $c$ and $d$ but they satisfy $c =_E d$. Finally, $a \rightarrow_A^1 q_0$ and $b \rightarrow_A^1 q_0$ but $a \rightarrow_A^1 q_0, b \rightarrow_A^1 q_1$ and $a \rightarrow_A^1 b$.

**Theorem 8 (Upper Bound [15]).** Let $\mathcal{R}$ be a left-linear TRS, $E$ a set of equations and $\mathcal{A}$ an $\mathcal{R}/E$-coherent automaton. For any $i \in \mathbb{N}$: $\mathcal{L}(\mathcal{A}_{\mathcal{R}, E}^i) \subseteq \mathcal{L}_{\mathcal{E}}(\mathcal{L}(\mathcal{A}))$ and $\mathcal{A}_{\mathcal{R}, E}^i$ is $\mathcal{R}/E$-coherent.

Finally, we state the termination theorem which relies on $E$-compatibility. Roughly speaking, $E$-compatibility is the symmetric of $E$-coherence. An automaton $\mathcal{A}$ is $E$-compatible if for all states $q_1, q_2 \in \mathcal{A}$ and all terms $s, t \in \mathcal{T}(\mathcal{F})$ such that $s \rightarrow_A^1 q_1, t \rightarrow_A^1 q_2$ and $s =_E t$ then we have $q_1 = q_2$.

**Theorem 9 (Termination of completion [10]).** Let $\mathcal{A}$ be a $\kappa$-reduced tree automaton, $\mathcal{R}$ a left-linear TRS, and $E$ a set of equations such that $\mathcal{T}(\mathcal{F})/\equiv_E$ is finite. If for all $i \in \mathbb{N}$, $\mathcal{A}_{\mathcal{R}, E}^i$ is $E$-compatible then there exists a natural number $k \in \mathbb{N}$ such that $\mathcal{A}_{\mathcal{R}, E}^k$ is a fixpoint.

To prove our final result, we first have to generalize Theorems 8 and 9 to discard the technical $\mathcal{R}/E$-coherence and $E$-compatibility assumptions. This is the objective of the next sections.

### 4 From automata to equations and vice versa

Theorem 9 uses the assumption that the automata $\mathcal{A}_{\mathcal{R}, E}^i$ are all $E$-compatible. This is not true in general. Unlike $\mathcal{R}/E$-coherence, $E$-compatibility is not preserved by tree automaton completion: $\mathcal{A}_{\mathcal{R}, E}^{i+1}$ may not be $E$-compatible even if $\mathcal{A}_{\mathcal{R}, E}^i$ is. Proofs can be found in [11].

**Example 10.** Let $\mathcal{F} = \{ f : 1, a : 0, b : 0, c : 0 \}$, $\mathcal{R} = \{f(x) \to f(f(x)), f(f(x)) \to a\}$, $\mathcal{A}$ be the automaton such that $\Delta = \{a \to q_1, c \to q_1, f(q_1) \to q_f\}$ and $E = \{f(a) = f(b), f(b) = b, f(c) = f(b)\}$. Note that $\mathcal{T}(\mathcal{F})/\equiv_E$ has 3 equivalence classes: the class of $\{a\}$, the class of $\{b, f(a), f(b), f(c), \ldots\}$ and the class of $\{c\}$. However, completion does not terminate on this example. Automaton $\mathcal{A}$ is $E$-compatible ($f(a) =_E f(c)$ and both terms are recognized with $\rightarrow_A^1$ by the same state: $q_f$) but $\mathcal{A}_{\mathcal{R}, E}^1$ is not: it has one new state $q_2$ and contains additional transitions ($f(q_f) \to q_2, q_2 \to q_f$). We thus have $f(f(a)) \rightarrow_{\mathcal{A}_{\mathcal{R}, E}^1} q_2$ and $f(a) \rightarrow_{\mathcal{A}_{\mathcal{R}, E}^1} q_f$ and $f(f(a)) =_E f(f(b)) =_E f(b) =_E f(a)$ but $q_2 \neq q_f$. Since $b$ is not recognized by $\mathcal{A}_{\mathcal{R}, E}^i$ for any $i$, the equation $f(b) = b$ never applies and completion diverges.

$E$-compatibility can be ensured for particular cases of $\mathcal{R}$ and $E$, e.g., for typed functional programs [10]. Here, we show how to transform the set $E$ into a set $E_{\mathcal{B}}$ for which completed automata are $E_{\mathcal{B}}$-compatible, and completion is thus terminating. We also build $E_{\mathcal{B}}$ so that its precision is similar to $E$, i.e., $=_{E_{\mathcal{B}}} \equiv =_{E_{\mathcal{B}}}$. This transformation is based on the Myhill-Nerode theorem for trees [18, 6]. We first produce a tree automaton $\mathcal{B}$ whose states recognize the equivalence classes of $E$. Then, from $\mathcal{B}$, we perform the inverse operation and obtain a set $E_{\mathcal{B}}$ whose set of equivalence classes is similar to the classes of $E$, but whose
equations avoid the problem shown in Example 10. In this paper, we mainly consider sets $E$ of ground equations because they are sufficient to prove our completeness results and for the practical applications of Section 7. However, this can be extended to general equations if $E$ can be oriented into a weakly terminating TRS $R$ s.t. $\text{IRR}(R)$ is finite [11].

### 4.1 From equations to automata

Provided that $T(F)/_{=E}$ is finite, the Myhill-Nerode theorem for trees [18, 6] relates $T(F)/_{=E}$ with tree automata. This theorem is constructive and provides an algorithm to switch from one form to the other, provided that $=E$ is decidable. In the following we denote by $\text{MN}$ the function that builds a tree automaton from a set of equations $E$ [18].

▶ **Definition 11 (Function MN).** Let $E$ be a set of equations such that $T(F)/_{=E}$ is finite and $=E$ is decidable. Let $Q$ be a set of states and $\text{state} : T(F)/_{=E} \rightarrow Q$ be an injective function. $\text{MN}(E) = (F, Q, Q, \Delta)$ where $\Delta = \{ (\text{state}(u_1), \ldots, \text{state}(u_n)) \rightarrow \text{state}(u) | f \in E, u_1, \ldots, u_n, u \in T(F)/_{=E} \text{ and } f(u_1, \ldots, u_n) =E u \}$

▶ **Theorem 12 (Myhill-Nerode theorem for trees [18]).** If $T(F)/_{=E}$ is finite and $=E$ decidable, $B = \text{MN}(E)$ is a reduced, deterministic, epsilon-free and complete tree automaton such that for all $s, t \in T(F), s =E t \iff (3q : \{ s, t \} \subseteq \mathcal{L}(B, q))$.

When all equations of $E$ are ground, $E$ can be oriented into a complete TRS (confluent and terminating) $\overline{E}$, using for instance [22]. Then $=E$ is decidable using $\overline{E}$ and finiteness of $T(F)/_{=E}$ is equivalent to finiteness of $\text{IRR}(\overline{E})$, which is decidable [6].

▶ **Example 13.** Consider the set $E$ of Example 10. We can orient $E$ into a complete TRS $\overline{E} = \{ f(a) \rightarrow f(b), f(b) \rightarrow b, f(c) \rightarrow f(b) \}$. The set $\text{IRR}(\overline{E})$ is $\{ a, b, c \}$. The automaton $\text{MN}(E)$ has 3 states $q_0, q_1, q_2$ such that $\text{state}(a) = q_0$, $\text{state}(b) = q_1$ and $\text{state}(c) = q_2$. It has six transitions $a \rightarrow q_0$ (because $a \rightarrow_{\overline{E}}^1 a$), $b \rightarrow q_1$ (because $b \rightarrow_{\overline{E}}^1 b$), $c \rightarrow q_2$ (because $c \rightarrow_{\overline{E}}^1 c$), $f(q_0) \rightarrow q_1$ (because $f(a) \rightarrow_{\overline{E}}^1 b$), $f(q_1) \rightarrow q_1$ (because $f(b) \rightarrow_{\overline{E}}^1 b$), $f(q_2) \rightarrow q_1$ (because $f(c) \rightarrow_{\overline{E}}^1 b$).

### 4.2 From automata to equations

In the other direction, starting from a tree automaton $B$ it is possible to build a set of equations $E_B$ such that languages recognized by states of $B$ and equivalence classes of $T(F)/_{=E_B}$ coincide [18]. We reformulate the original algorithm into a function called $A2E$ because we need some additional properties on the generated set of equations for completion to terminate. For simplicity we assume that $B$ is Reduced and epsilon-Free. Some properties of $E_B$ will hold only if $B$ is also Complete and Deterministic. In the following, we use the $\text{RF}$ and $\text{RDFC}$ short-hands for automata having the related properties. Recall that for any tree automaton, there exists an equivalent RF or RDFC automaton [6].

For an RF automaton $B$, the construction of $E_B = A2E(B)$ is straightforward and follows [18]: for all states $q$ we identify a ground term recognized by $q$, a representative, and for all transitions $f(q_1, \ldots, q_n) \rightarrow q$ we generate an equation $f(t_1, \ldots, t_n) = t$ where $t_i, 1 \leq i \leq n$ are representatives for $q_i$ and $t$ is a representative for $q$. However, for this set of equations to guarantee termination of completion it needs some redundancy: for each state we generate a set of state representatives and the equations are defined for each representative of the set. As shown in Example 10, the equation $f(b) = b$ cannot be applied during completion because $b$ does not occur in the tree automaton. However, a logical consequence of this equation is that $f(f(a)) =E f(a)$ and terms $f(f(a))$ and $f(a)$
that occur in the tree automaton could be merged. In our setting the term $f(a)$ will be
a state representative and the equation $f(f(a)) = f(a)$ will appear in the set of generated
equations. Roughly speaking, every constant symbol $a$ appearing in a transition $a \rightarrow q$ is a
state representative for $q$. Every term of the form $f(u_1, \ldots, u_n)$ is a state representative for
$q$ if (1) $u_i$’s are not state representatives of $q$, (2) $f(q_1, \ldots, q_n) \rightarrow q$ is a transition of $B$ and
(3) $u_i$’s are state representatives for the $q_i$’s. The property (1) ensures finiteness of the set of
representatives.

**Definition 14 (State representatives).** Let $B = (\mathcal{F}, Q, Q_f, \Delta)$ be an RF tree automaton
and $q \in Q$. The set of state representatives of $q$ of height lesser or equal to $k \in \mathbb{N}$, denoted
by $[q]_B^k$, is inductively defined by:

$$
[q]_B^0 = \{a \mid a \rightarrow q \in \Delta\}
$$

$$
[q]_B^1 = [q]_B^{k-1} \cup \{f(u_1, \ldots, u_n) \mid f(q_1, \ldots, q_n) \rightarrow q \in \Delta \text{ and } \forall i \in \{1, \ldots, n\} : u_i \in [q_i]_B^{k-1},
\text{ and } \forall p \in \text{Pos}(u_i) : u_i[p] \not\in [q]_B^{k-1}\}
$$

In the above definition, the fact that $B$ is reduced and epsilon-free ensures that there exists
at least one (non-epsilon) transition for every state and that each state has at least one state
representative.

**Example 15.** Let $B$ be the RF automaton that we obtained in Example 13 and whose
set of transitions is $a \rightarrow q_0$, $b \rightarrow q_1$, $c \rightarrow q_2$, $f(q_0) \rightarrow q_1$, $f(q_1) \rightarrow q_1$, $f(q_2) \rightarrow q_1$.

$$
[q_0]_B^1 = \{a\}, [q_1]_B^1 = \{b\}, \text{ and } [q_2]_B^1 = \{c\}.
$$

$$
[q_0]_B^2 = [q_0]_B^1, [q_1]_B^2 = \{b, f(a), f(c)\}, \text{ and } [q_2]_B^2 = [q_2]_B^1. \text{ The term } f(b) \text{ of height 2 and}
\text{ recognized by } q_1 \text{ is not added to } [q_1]_B^2 \text{ because its subterm } b \text{ belongs to } [q_1]_B^1.
$$

$$
\text{The fixpoint is reached because terms } f(f(a)) \text{ and } f(f(c)) \text{ recognized by } q_1 \text{ are not added to}
[q_1]_B^2 \text{ because } f(a) \text{ and } f(c) \text{ belong to } [q_1]_B^2.
$$

We denote by $[q]_B$ the set of all state representatives for the state $q$ i.e., the fixpoint of
the above equations. We know that such a fixpoint exists and is always a finite set. Omitted
proofs can be found in [11].

**Lemma 16 (The set of state representatives is finite).** For all RF tree automata $B$, for all
states $q \in B$ there exists a natural number $k \in \mathbb{N}$ for which the set $[q]_B^k$ is a fixpoint.

**Definition 17 (Function $A2E$: set of equations $E_B$ from a tree automaton $B$).** Let $B =
(\mathcal{F}, Q, Q_f, \Delta)$ be an RF tree automaton. The set of equations $E_B$ inferred from $B$ is
$A2E(B) = \{f(u_1, \ldots, u_n) = u \mid f(q_1, \ldots, q_n) \rightarrow q \in B, u \in [q]_B \text{ and } u_i \in [q_i]_B \text{ for } 1 \leq i \leq n\}$.

**Example 18.** Starting from the automaton $B$ and the state representatives of Example 15,
the set $A2E(B)$ contains the following equations: $a = a$ (because of transition $a \rightarrow q_0$), $c = c$
(because of transition $c \rightarrow q_2$), $b = b$, $b = f(a)$, $b = f(c)$ (because of transition $b \rightarrow q_1$),
$f(a) = f(a)$, $f(a) = b$, $f(a) = f(c)$ (because of transition $f(q_0) \rightarrow q_1$), $f(f(a)) = f(a)$,
$f(f(a)) = b$, $f(f(a)) = f(c)$, $f(b) = f(a)$, $f(b) = b$, $f(b) = f(c)$, $f(f(c)) = f(a)$, $f(f(c)) = b$,
$f(f(c)) = f(c)$ (because of transition $f(q_1) \rightarrow q_1$), $f(c) = f(a)$, $f(c) = b$, and $f(c) = f(c)$
(because of transition $f(q_2) \rightarrow q_1$).

Since $B$ is finite and the set of state representatives is finite then so is $E_B$. Note that many
equations of $E_B$ are useless w.r.t. the underlying equational theory. This is the case, in the
above example, for equations of the form $a = a$ as well as the equation $f(a) = f(c)$ which
is redundant w.r.t. $b = f(a)$ and $b = f(c)$. However, as shown in Example 10 those equations are necessary for equational simplification to produce $E_B$-compatible automata and completion to terminate. With the above $E_B$, completion of Example 10 terminates. Below, Theorem 23 shows that, if $B$ is RDFC then completion with $A2E(B)$ always terminates. Unsurprisingly, if $B$ is deterministic then equivalence classes of $E_B$ coincide with languages recognized by states of $B$. This is the purpose of the next two lemmas.

**Lemma 19.** Let $B = \langle F, Q, Q_f, \Delta \rangle$ be an RDFC tree automaton and $E_B = A2E(B)$. For all $s \in T(F)$, there exists a unique state $q \in Q$ such that $s \rightarrow^*_B q$ and for all state representatives $u \in [q]_B$, $s =_{E_B} u$.

Now we can relate equivalence classes of $E_B$ and languages recognized by states of $B$.

**Lemma 20** (Equivalence classes of $E_B$ coincide with languages recognized by states of $B$). Let $B = \langle F, Q, Q_f, \Delta \rangle$ be an RDFC tree automaton and $E_B = A2E(B)$. For all states $s, t \in T(F)$, $s =_{E_B} t \iff (\exists q : \{s, t\} \subseteq L(B, q))$.

**Corollary 21** ($T(F)/_{E_B}$ is finite). Let $B = \langle F, Q, Q_f, \Delta \rangle$ be an RDFC tree automaton. If $E_B$ is the set of equations inferred from $B$ then $T(F)/_{E_B}$ is finite.

## 5 Generalizing the termination theorem

Now, we prove that using $E_B$ built from an RDFC tree automaton $B$, completion terminates.

### 5.1 Proving termination of completion with $E_B$

In the following, the automaton $A^*$ is the limit of the (possibly) infinite completion of an initial $\kappa$-reduced tree automaton $A$ with $R$ and $E_B$. If the initial automaton is not $\kappa$-reduced then completion may diverge. For instance, completion of the automaton whose set of transitions is $\{f(q_0) \rightarrow q_1\}$, with $R = \{f(x) \rightarrow f(f(x))\}$ and $E = \{f(a) = a\}$ diverges (simplification never happens because $q_0$ does not recognize any term). Now we show that all state representatives are recognized by epsilon-free derivations in $A^*$.

**Lemma 22** (All states of $A^*$ recognize at least one state representative). Let $R$ be a TRS, $A$ a $\kappa$-reduced tree automaton, $B$ an RDFC tree automaton and $E_B = A2E(B)$. Let $A^*$ be the limit of the completion of $A$ by $R$ and $E_B$. For all states $q \in A^*$, for all terms $s \in T(F)$ such that $s \rightarrow^*_A q$, there exists a state $q'_B \in B$, a term $u \in [q'_B]_B$ such that $u =_{E_B} s$ and $u \rightarrow^*_{A^*} q$.

Now, we can state the termination theorem with $E_B$.

**Theorem 23** (Completion with $E_B$ terminates). Let $R$ be a TRS, $A$ a $\kappa$-reduced tree automaton, $B$ be an RDFC tree automaton and $E_B = A2E(B)$. Let $n$ be the number of all states representatives of $B$. The automaton $A^*$, limit of the completion of $A$ with $R$ and $E_B$, has $n$ states or less.

**Proof.** Recall that the number $n$ of state representatives is finite (cf. Lemma 16). Assume that $A^*$ has $m$ distinct states with $m > n$. From Lemma 22 we know that for all states $q \in A^*$, there exists a state representative $u$ such that $u \rightarrow^*_{A^*} q$. Since there are only $n$ state representatives, by pigeon hole principle, we know that there is necessarily one state representative $u$ recognized by two distinct states $q_1$ and $q_2$ of $A^*$. Thus, $u \rightarrow^*_{A^*} q_1$ and $u \rightarrow^*_{A^*} q_2$. Besides, by construction of $E_B$, we know that the equation $u = u$ is part of $E_B$. This contradicts the fact that $A^*$ is simplified w.r.t. $E_B$. ▶
5.2 Building $E_B$ from any set of equations $E$

Now, we combine the transformations $A2E$ and $MN$ to produce a set of equations $E_B$ that ensures termination of completion. Unsurprisingly, $E_B$ is equivalent to $E$.

Lemma 24. Let $E$ be a set of equations. If $T(F)/=_{E}$ is finite and $=_{E}$ is decidable then $E_B = A2E(MN(E))$ and $=_{E} \equiv =_{E_B}$.

Theorem 25 (Generalized termination theorem for completion). Let $E$ be a set of ground equations such that $T(F)/=_{E}$ is finite. For all $\chi$-reduced tree automata $A$ and TRSs $R$, completion of $A$ with $R$ and $A2E(MN(E))$ terminates.

Proof. As mentioned in Section 4.1, since $E$ is ground $=_{E}$ is decidable. By Theorem 12, we know that $B = MN(E)$ exists and is RD FC. Let $E_B$ be the set of equations $A2E(B)$. Using Theorem 23, we know that completion of $A$ with $R$ and $E_B$ is terminating. ▷

The above theorem shows how to tune a set of equations $E$ into $E_B$ to guarantee termination of completion. Note that tuning $E$ into $E_B$ does not jeopardize the precision of the completion since Lemma 24 guarantees that $=_{E} \equiv =_{E_B}$. Combining this lemma with Theorem 8 (the Upper Bound Theorem) yields that completion of $R$ with $E_B$ is upper-bounded by $R_{E_B}$.

6 Improving the Precision of Equational completion

Looking at our overall goal, we are half way there. If $L$ is regular and $L \supseteq R^+(L(A))$ (or $L = R^+(L(A))$) then it can be recognized by an automaton $B$. Using the results of the last section, we can build a set of equations $E_B$ guaranteeing termination of completion. What remains to be proved is that completion with $E_B$ ends on a tree automaton underapproximating $L$ (or recognizing exactly $L = R^+(L(A))$). As it is, Theorem 8 (the Upper Bound Theorem) fails to tackle this goal because it needs $R/E$-coherence of $A$. However, if $A$ is not $R/E$-coherent the full precision, granted by this theorem, may not be obtained.

Example 26. Starting from Example 10, together with the set of equations $E_B$ of Example 18, the initial tree automaton is not $R/E_B$-coherent (nor $R/E$-coherent): $a \rightarrow \lambda^*_A q_1$ and $c \rightarrow \lambda_A q_1$ though $a \neq E_B c$. As a consequence, if we complete $A$ with $R$ and $E_B$, we obtain an automaton that roughly approximates $R^+(L(A))$. This can be done using the Timbuk tool [12]:

States $q_0$ $q_1$ Final States $q_0$ Transitions $c \rightarrow q_1$ $a \rightarrow q_1$ $c \rightarrow q_0$ $f(q_0) \rightarrow q_0$ $f(q_1) \rightarrow q_0$ $a \rightarrow q_0$

This automaton recognizes the term $c$ that is not reachable by rewriting the initial language $L(A) = \{f(a), f(c)\}$ with $R$ (nor by rewriting with $R/E_B$). We propose to transform $A$ so that it becomes $R/E$-coherent: we build the product between $A$ and $MN(E)$. We recall the definition of a product automaton and we show that the product is $R/E$-coherent.

Definition 27 (Product automaton [6]). Let $A = (F, Q, Q_F, \Delta_A)$ and $B = (F, P, P_F, \Delta_B)$ be automata. The product of $A$ and $B$ is $A \times B = (F, Q \times P, Q_F \times P_F, \Delta)$ where $\Delta = \{f((q_1, p_1), \ldots, (q_k, p_k)) \rightarrow (q', p') \mid f(q_1, \ldots, q_k) \rightarrow q' \in \Delta_A \text{ and } f(p_1, \ldots, p_k) \rightarrow p' \in \Delta_B\}$.

Theorem 28 (Generalized Upper Bound). Let $\mathcal{R}$ be a left-linear TRS, $A$ an epsilon-free automaton, and $E$ a set of ground equations such that $T(F)/=_{E}$ is finite. If $B = MN(E)$ and $\mathcal{A} = A \times B$ then for any $i \in \mathbb{N}$: $L(\mathcal{A}_{R,E}) \subseteq R_{E_B}^+(L(A))$. 
Lemma 30. We first need a lemma showing that if $\mathcal{R}$ is a TRS, then completion can build it. Since the upper-bound of completion depends on $\mathcal{R}$, we only have to prove that for all states $q$ of $\mathcal{A}$ and for all two terms $s, t \in \mathcal{T}$ such that (1) $s \rightarrow^{*}_A q$ and (2) $t \rightarrow^{*}_A q$ then $s =_E t$. Since $\mathcal{A}$ is a product automaton, $q$ is a pair of the form $(q_1, q_2)$ where $q_1 \in \mathcal{A}$ and $q_2 \in \mathcal{B}$. From (1) and (2) we can deduce that $s \rightarrow^{*}_B q_2$ and $t \rightarrow^{*}_B q_2$. Then, using Lemma 12, we get $s =_E t$. Thus $\mathcal{A}$ is $\mathcal{R}$-coherent and from Theorem 8, we get that $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{R}^*_E(\mathcal{L}(\mathcal{A}))$ and $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A})$ ends the proof.

Example 29. Starting from Example 26, we can build the product between $\mathcal{A}$ and the automaton $\mathcal{B}$ found in Example 13. In $\mathcal{A} \times \mathcal{B}$, $a$ and $c$ are recognized by two different states, avoiding the $\mathcal{R}/E$-coherence problem of Example 26. The $\kappa$-reduced product $\mathcal{A} = \mathcal{A} \times \mathcal{B}$ (where product states are renamed) is the automaton with $Q_f = \{q_2\}$ and $\Delta = \{c \rightarrow q_0, a \rightarrow q_1, f(q_0) \rightarrow q_2, f(q_1) \rightarrow q_2\}$. Running Timbuk on $\mathcal{A}$, $\mathcal{R}$, and $\mathcal{E}_B$, we obtain $\mathcal{A}^*_K.E$ whose precision is now bounded by $\mathcal{R}^*_E(\mathcal{L}(\mathcal{A}))$ and does not recognize $c$ in a final state:

States $q_0 \ q_1 \ q_2$ Final States $q_0$ Transitions $a \rightarrow q_1$ $f(q_0) \rightarrow q_0$ $f(q_1) \rightarrow q_0$ $f(q_2) \rightarrow q_0$ $a \rightarrow q_0$ $c \rightarrow q_2$

This provides hints to define equations for completion: we can start from an automaton $\mathcal{B}$ defining a rough approximation of the target language and build $E = \mathcal{A}2\mathcal{E}(\mathcal{B})$. Then, we complete $\mathcal{A} = \mathcal{A} \times \mathcal{B}$ with $\mathcal{R}$ and $\mathcal{E}$ and obtain a tree automaton $\mathcal{A}^*_K.E$ whose precision is better or equal to $\mathcal{B}$. The set $\mathcal{R}^*_E(\mathcal{L}(\mathcal{A}))$ acts as a safeguard for completion (see Figure 4). In particular, terms of $\mathcal{R}^*_E(\mathcal{L}(\mathcal{A}))$ may not belong to $\mathcal{L}(\mathcal{A})$. This is the case in Example 29, where the term $b$ belongs to $\mathcal{R}^*_E(\mathcal{L}(\mathcal{A}))$ but not to $\mathcal{L}(\mathcal{A})$. In practice, we still need to know if $E$ always exists (Section 7) and to generate a satisfactory $E$ (Section 8).

7 Completeness Theorems

In this section, we prove two completeness theorems on completion. The first theorem states that if the set of reachable terms can be over-approximated by a regular language $L$, then we can find a language containing reachable terms and under-approximating $L$ using equational completion. The second theorem states that if the set of reachable terms is regular then completion can build it. Since the upper-bound of completion depends on $\mathcal{R}^*_E$, we first need a lemma showing that if $E$ is built from $L$ then $\mathcal{R}^*_E$ is upper-bounded by $L$.

Lemma 30. Let $\mathcal{R}$ be a TRS over $\mathcal{F}$, $S \subseteq \mathcal{T}$, and $\mathcal{B}$ an RDFS automaton such that $\mathcal{L}(\mathcal{B}) \supseteq \mathcal{R}^*_E(S)$ and $\mathcal{L}(\mathcal{B})$ is $\mathcal{R}$-closed. If $E_B = \mathcal{A}2\mathcal{E}(\mathcal{B})$ then $\mathcal{R}^*_E(S) \subseteq \mathcal{L}(\mathcal{B})$.

Example 31 shows that the $\mathcal{R}$-closed assumption on $L$ is necessary for the lemma to hold.
Completeness of Tree Automata Completion

Example 31. Let \( \mathcal{F} = \{ a : 0, b : 0, c : 0, d : 0 \} \), \( S = \{ a \} \), \( \mathcal{R} = \{ a \to b, c \to d \} \), and \( \mathcal{L} = \{ a, b, c \} \) where \( \mathcal{L} \supseteq \mathcal{R}^*(S) \) but \( \mathcal{L} \) is not \( \mathcal{R} \)-closed. A possible RDFC automaton \( \mathcal{B} \), s.t. \( \mathcal{L}(\mathcal{B}) = \mathcal{L} \), has a unique final state \( q \) and transitions \( \{ a \to q, b \to q, c \to q \} \). Thus \( E_B = \text{A2E}(\mathcal{B}) \) includes the equation \( b = c \). Finally \( \mathcal{R}^*_E(S) = \{ a, b, c, d \} \not\subseteq \mathcal{L} \).

Theorem 32 (Completeness). Let \( \mathcal{A} \) be a reduced epsilon-free tree automaton and \( \mathcal{R} \) a left-linear TRS. Let \( \mathcal{T}(\mathcal{F}) \supseteq \mathcal{L} \supseteq \mathcal{R}^* (\mathcal{L}(\mathcal{A})) \). If \( \mathcal{L} \) is regular and \( \mathcal{R} \)-closed then there exists a set of ground equations \( E \) such that \( \mathcal{A} = \mathcal{A} \times \text{MN}(E) \), \( \mathcal{A}^*_{R,E} \) exists and \( \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A}^*_{R,E}) \subseteq \mathcal{L} \).

Proof. Since \( \mathcal{L} \) is regular, we know that there exists an RDFC tree automaton, say \( \mathcal{B} \), recognizing \( \mathcal{L} \). From \( \mathcal{B} \) we can infer \( E_B = \text{A2E}(\mathcal{B}) \) and then use completion to compute reachable terms. From Theorem 23, we know that completion of the automaton \( \mathcal{A} \) with \( \mathcal{R} \) and the set of equations \( E_B \) always terminates on a tree automaton \( \mathcal{A}^*_{R,E} \). From Theorem 8, we know that \( \mathcal{L}(\mathcal{A}^*_{R,E}) \subseteq \mathcal{R}^*_E (\mathcal{L}(\mathcal{A})) \) provided that \( \mathcal{A} \) is \( \mathcal{R}/E_B \)-coherent. To enforce \( \mathcal{R}/E_B \)-coherence of \( \mathcal{A} \), we apply the transformation presented in Section 6. Let \( \mathcal{A} = \mathcal{A} \times \text{MN}(E_B) \). Note that since \( E_B \) is obtained by using the \( \text{A2E} \) transformation, \( \mathcal{T}(\mathcal{F})/_{=E} \) is finite (Corollary 21) and since equations of \( E_B \) are ground, \( =_{E_B} \) is decidable. The resulting automaton \( \mathcal{A} \) is \( \mathcal{R}/E_B \)-coherent. Besides, Theorem 23 also applies to \( \mathcal{A} \). Thus, completion of \( \mathcal{A} \) with \( \mathcal{R} \) and \( E_B \) always ends on an automaton \( \mathcal{A}^*_{R,E} \). The automaton \( \mathcal{A}^*_{R,E} \) satisfies \( \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A}^*_{R,E}) \) (by Theorem 5) and \( \mathcal{L}(\mathcal{A}^*_{R,E}) \subseteq \mathcal{R}^*_E (\mathcal{L}(\mathcal{A})) \) (by Theorem 28). Since \( \mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}^*_{R,E}) \), we have \( \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A}) \) and \( \mathcal{L}(\mathcal{A}) \subseteq \mathcal{R}^*_E (\mathcal{L}(\mathcal{A})) \). With Lemma 30, we get that \( \mathcal{R}^*_E (\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{B}) = \mathcal{L} \).

In general we do not have \( \mathcal{L}(\mathcal{A}^*_{R,E}) \supseteq \mathcal{L} \) because \( \mathcal{L}(\mathcal{A}^*_{R,E}) \) can be more precise than \( \mathcal{L} \) (See Example 29). However, this is true when \( \mathcal{L} = \mathcal{R}^* (\mathcal{L}(\mathcal{A})) \), as we show in the next theorem.

Theorem 33 (Completeness for regularity preserving TRSs). Let \( \mathcal{A} \) be a reduced epsilon-free tree automaton and \( \mathcal{R} \) a left-linear TRS. If \( \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \) is regular then it is possible to compute a tree automaton recognizing \( \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \) by equational tree automata completion.

Proof. Let \( \mathcal{L} = \mathcal{R}^* (\mathcal{L}(\mathcal{A})) \). It is \( \mathcal{R} \)-closed. By assumption, it is also regular. Thus, we can apply Theorem 32 to get that there exists a set of equations \( E \) and a tree automaton \( \mathcal{A} = \mathcal{A} \times \text{MN}(E) \) such that \( \mathcal{A}^*_{R,E} \) exists and \( \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A}^*_{R,E}) \subseteq \mathcal{L} \). Since \( \mathcal{L} = \mathcal{R}^* (\mathcal{L}(\mathcal{A})) \), we get \( \mathcal{L}(\mathcal{A}^*_{R,E}) = \mathcal{R}^* (\mathcal{L}(\mathcal{A})) \).

Thus, completion is complete w.r.t. all left-linear TRS classes preserving regularity.

Application of the Completeness Theorem

Let us show how to take advantage of Theorem 32 to automatically verify safety properties on programs. Given an initial regular language \( S \) and a program represented by a TRS \( \mathcal{R} \), we can prove that the program never reaches terms in a set \( \text{Bad} \) by checking that there exists a regular over-approximation \( \mathcal{L} \supseteq \mathcal{R}^* (S) \) such that \( \mathcal{L} \cap \text{Bad} = \emptyset \). This technique has been used to verify cryptographic protocols [1], Java programs [4] and Functional Programs [10, 14]. Theorem 32 ensures that, if there exists an \( \mathcal{R} \)-closed regular approximation \( \mathcal{L} \) such that \( \mathcal{L} \cap \text{Bad} = \emptyset \), then we can build it (or under-approximate it) using completion and an appropriate set of ground equations \( E \). To explore all the possible \( E \), it is enough to explore \( G_T(k) \) with \( k \in \mathbb{N}^* \).
Definition 34 (Generated Equations for \( F \) and \( k \in \mathbb{N}^* \)). Let \( \mathcal{B}(k) \) be the set of all possible RDFC tree automata on \( F \) with exactly \( k \) states. The set of generated equations of size \( k \) is \( G_F(k) = \{ E \mid \mathcal{B} \in \mathcal{B}(k) \text{ and } E = A2E(B) \} \).

The semi-algorithm to prove that \( \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \cap Bad = \emptyset \) works as follows: (a) We start from \( k = 1 \), (b) we generate \( G_F(k) \), (c) we try completion with \( \mathcal{A}, \mathcal{R} \) and all \( E \in G_F(k) \) (completion terminates with all those \( E \), Theorem 23). If \( \mathcal{L}(\mathcal{A}_{\mathcal{R},E}^*) \cap Bad = \emptyset \) for one \( E \), we are done. Otherwise if \( \mathcal{L}(\mathcal{A}_{\mathcal{R},E}^*) \cap Bad \neq \emptyset \) for all \( E \in G_F(k) \), we increase \( k \) and go back to step (b). If there exists a regular over-approximation \( \mathcal{L} \supseteq \mathcal{R}^*(S) \) such that \( \mathcal{L} \cap Bad = \emptyset \), then this algorithm eventually reaches a tree automaton \( \mathcal{B} \) such that \( \mathcal{L}(\mathcal{B}) = \mathcal{L} \), \( E = A2E(B) \), and by Theorem 32, we know that \( \mathcal{L}(\mathcal{A}_{\mathcal{R},E}^*) \subseteq \mathcal{L} \). Finally, since \( \mathcal{L} \cap Bad = \emptyset \), we have \( \mathcal{L}(\mathcal{A}_{\mathcal{R},E}^*) \cap Bad = \emptyset \).

For general TRSs, we can enumerate all equation sets of \( G_F(k) \) but the search space is huge. When the TRS \( \mathcal{R} \) encodes a functional program, we can restrict the search space to equation sets of the form \( E = E_R \cup E_r \cup E_C \) [10], where \( E_R \) and \( E_r \) are fixed and \( E_C \) only ranges over \( \text{IRR}(\mathcal{R}) \). If program’s functions are complete and terminating, \( \text{IRR}(\mathcal{R}) \) is the set of constructor terms. The set \( \mathcal{F} \) can be separated into a set of defined symbols \( \mathcal{D} = \{ f \mid \exists r \in \mathcal{R} \text{ s.t. } \mathcal{R} \text{oot}(l) = f \} \) and constructor symbols \( \mathcal{C} = \mathcal{F} \setminus \mathcal{D} \).

Definition 35 (\( E_r \)). For a given set of symbols \( \mathcal{F} \), \( E_r = \{ f(x_1, \ldots, x_n) = f(x_1, \ldots, x_n) \mid f \in \mathcal{F} \text{, and arity of } f \text{ is } n \} \), where \( x_1 \ldots x_n \) are pairwise distinct variables.

Definition 36 (\( E_R \)). Let \( \mathcal{R} \) be a TRS, the set of \( \mathcal{R} \)-equations is \( E_R = \{ l = r \mid l \rightarrow r \in \mathcal{R} \} \).

Definition 37 (\( E_C \) contracting equations for \( \mathcal{T}(\mathcal{C}) \)). A set of equations is contracting for \( \mathcal{T}(\mathcal{C}) \), denoted by \( E_C \), if all equations of \( E_C \) are of the form \( u = u \mid_p \) with \( u \in \mathcal{T}(\mathcal{C}) \), \( p \neq \lambda \), \( E_C^* = \{ u \rightarrow u \mid_p \mid u \in E_C \} \), and \( \text{IrIr}(E_C^*) \) (terms of \( \mathcal{T}(\mathcal{C}) \) irreducible by \( E_C^* \)) is finite.

Completion is terminating if \( E = E_R \cup E_r \cup E_C \) and \( \mathcal{R} \) encodes a functional program which is terminating, complete, and is either first order [10] or higher-order [14]. Now, our objective is to define a completeness theorem for TRSs encoding those programs. Since \( E_C \) contains \( E_R \), all completed automata \( \mathcal{A}_{\mathcal{R},E}^* \) will be \( \mathcal{R} \)-closed because \( s \rightarrow s \), \( s \rightarrow t \cdot q \rightarrow q \rightarrow q' \) implies that \( s = t \cdot q \rightarrow q' \) (\( \mathcal{A}_{\mathcal{R},E}^* \) is simplified w.r.t. \( \mathcal{R} \supseteq E_R \)). Thus, the completeness theorem says that if there exists an \( \mathcal{R} \)-closed automaton \( \mathcal{B} \) s.t. \( \mathcal{L}(\mathcal{B}) \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \) then there exists \( E_C \) such that \( E = E_R \cup E_r \cup E_C \) and \( \mathcal{L}(\mathcal{A}_{\mathcal{R},E}^*) \subseteq \mathcal{L}(\mathcal{B}) \). To prove such a theorem, we need to explain how to construct a satisfying \( E_C \) from \( \mathcal{B} \). We propose to project \( \mathcal{B} \) on \( \mathcal{C} \) (denoted by \( \mathcal{B}/\mathcal{C} \)), produce equations from \( \mathcal{B}/\mathcal{C} \) with \( A2E \), and finally filter out all equations that are not of the form \( u = u \mid_p \) (this is function \( \mathcal{E} \)).

Definition 38 (Automaton projection on \( \mathcal{C} \)). Let \( \mathcal{B} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta) \) be an epsilon free tree automaton. The automaton \( \mathcal{B}/\mathcal{C} \) is the tree automaton \( (\mathcal{C}, \mathcal{Q}_R \cup \mathcal{Q}_C, \Delta_C) \) where \( \Delta_C = \{ s \rightarrow q \mid s \rightarrow q \in \Delta \land \mathcal{R} \text{oot}(s) \in \mathcal{C} \} \) and \( \mathcal{Q}_C \) is the set of states occurring in the right-hand side of transitions of \( \Delta_C \).

Note that \( \mathcal{L}(\mathcal{B}/\mathcal{C}) = \mathcal{L}(\mathcal{B}) \cap \mathcal{T}(\mathcal{C}) \) and if \( \mathcal{B} \) is RDFC so is \( \mathcal{B}/\mathcal{C} \). In particular, if \( \mathcal{B} \) is complete for \( \mathcal{F} \), \( \mathcal{B}/\mathcal{C} \) is complete for \( \mathcal{C} \).

Definition 39. Given a set of equations \( E \), \( \mathcal{E}(E) = \{ l = r \in E \mid r = l \mid_p \text{ and } p \neq \lambda \} \).

In the following, we show that \( E = \mathcal{E}(A2E(B)) \) is a contracting set of equations, provided that \( \mathcal{B} \) is RDFC. In particular, we show that \( \text{IRR}(E) \) is finite.
Lemma 40. If $B$ is an RDFC automaton on $C$ and $E = ct(A2E(B))$, then $\text{IRR}(\overrightarrow{E})$ is finite and $E$ is contracting for $T(C)$.

The above lemma states that $ct(A2E(B))$ is contracting for $T(C)$. To have a finite set of equivalence classes on $T(F)$ (and a terminating completion) we use $E = E_R \cup E_r \cup E_C$ where $E_C = ct(A2E(B/C))$. Now we prove that, w.r.t. approximations, $E$ is as precise as $E_B$.

Lemma 41. For a TRS $R$ and an automaton $B$ on $F$, if $B$ is RDFC and $R$-closed and $E_B = A2E(B)$, $E_C = ct(A2E(B/C))$, and $E = E_R \cup E_r \cup E_C$ then $=E \subseteq =E_B$.

Theorem 42 (($E_R \cup E_r \cup E_C$ covers all $R$-closed approximation automata). Let $R$ be a left-linear TRS and $A$ a reduced and epsilon-free tree automaton on $F$. Let $B$ be an $R$-closed RDFC tree automaton such that $L(B) \supseteq R^*(L(A))$. Let $E_C = ct(A2E(B/C))$, $E = E_R \cup E_r \cup E_C$, and $B = A \times MN(E)$. If $A_{R,E}$ exists then $R^*(L(A)) \subseteq L(A_{R,E}) \subseteq L(B)$.

Proof. The fact that $R^*(L(A)) \subseteq L(A_{R,E})$ is ensured by Theorem 5. Using the Generalized Upper Bound theorem (Theorem 28), we deduce that (1) $L(A_{R,E}) \subseteq L(R(E))$. From Lemma 41, we know that $=E \subseteq =E_B$ and thus that (2) $R^*_E(L(A)) \subseteq R^*_E(L(A))$. Besides, since $B$ is $R$-closed, $L(B)$ is $R$-closed and we can use Lemma 30 to get that (3) $R_E^*(L(A)) \subseteq L(B)$. Finally, using transitivity of $\subseteq$ on (1), (2) and (3) we get $L(A_{R,E}) \subseteq L(B)$.

Note that, for functional programs classes of [10] and [14], since $E_C = ct(A2E(B/C))$ is contracting (Lemma 40), $A_{R,E}$ always exists. Thus, if there exists an $R$-closed tree automaton $B$ such that $L(B) \supseteq R^*(L(A))$ and $L(B) \cap \text{Bad} = \emptyset$, it is enough to enumerate all possible $E = E_R \cup E_r \cup E_C$ to find it. Since $E_R$ and $E_r$ are fixed, it is enough to enumerate all possible $E_C$ on $C$ using Definition 37 and the algorithm of Definition 34 (generating on $C$).

Example 43. Let $C = \{0 : 0, s : 1\}$. For $k = 1$, there is only one RDFC automaton with 1 state. Its transitions are $\{s(q_0) \rightarrow q_0, 0 \rightarrow q_0\}$. Thus, $G_C(1) = \{s(0) = 0\}$. For $k = 2$ there are 2 RDFC automata: one with transitions $\{0 \rightarrow q_0, s(q_0) \rightarrow q_1, s(q_1) \rightarrow q_1\}$ and the other with transitions $\{0 \rightarrow q_0, s(q_0) \rightarrow q_1, s(q_1) \rightarrow q_0\}$. Thus, $G_C(2) = \{s(0) = s(0), s(s(0)) = 0, s(s(s(0))) = 0\}$.

We implemented this in Timbuk and used it to verify more than 20 safety properties of several first-order and higher-order functions on lists, ordered lists, trees and ordered trees. Higher-order properties include state-of-the-art examples from [21, 19, 14]. In [14], contracting equations of $E_C$ contain variables and are generated from test sets. Here, we generate ground contracting equations $E_C$ as shown above and use $E = E_R \cup E_r \cup E_C$ for completion. We transform the initial automaton $A$ into $\overrightarrow{A}$ as in Theorem 28. The approximation is, thus, upper-bounded by $R_E^*$ and we can benefit from the coverage guarantee of Theorem 42. On examples taken from [21, 19], we managed to do the same proofs with comparable execution times. On all the examples of [14], we do the same proofs (or find the counter-examples, see [14]), but in a much faster way. Appendix A presents a summary of those experiments and full details are here: http://people.irisa.fr/Thomas.Genet/timbuk/funExperiments/. For each example, we provide the specifications, Timbuk output, and the full result with completed automaton and generated equations in a Coq checkable file comp.res.

9 Conclusion and perspectives

Tree automata completion is known to cover many TRS classes preserving regularity [8, 10]. For some other classes, the question was still open. We established that, for all those classes...
(including those not known yet), given $A$ and $R$, there exists a set of equations $E$ such that $A_{R,E}$ recognizes $\mathcal{R}^*(\mathcal{L}(A))$. We proved a similar theorem for the approximated case. The proofs are not constructive but give hints to enumerate sets of equations $E$. Finally, we showed that if a regular approximation satisfying a given property exists, we can find it by enumerating the sets $E$ and running completion. From an algorithmic point of view and in the general case (where $\mathcal{T}(\mathcal{F})/\}_R$ is finite), since we enumerate tree automata $B$ on $\mathcal{T}(\mathcal{F})$ to generate sets of equations $E$, we could directly take advantage of $B$ to perform automata simplification and thus replace equations.

However, equations are strictly more powerful than tree automata to define approximations. This can be observed on functional programs (Section 8) where simplification and thus replace equations. This can be observed on functional programs (Section 8) where $R$ is generated using a tree automaton) but $\mathcal{T}(\mathcal{F})/\}_R$ is not [14] and $E$ cannot be defined with an automaton. On functional programs, Theorem 42 shows that enumeration can be restricted to sets of ground contracting equations on constructor symbols. This makes enumeration efficient enough to automatically verify properties on first-order and higher programs. Experiments shows that this approach tackles state-of-the-art automatic verification problems for first-order and higher-order programs. The completeness Theorem for functional programs ranges over $\mathcal{R}$-closed RDFC approximation automata. However, there exist $\mathcal{R}$-closed approximations that are not recognized by $\mathcal{R}$-closed RDFC tree automata.

Example 44. Let $\mathcal{F} = \{ f : 1, a : 0, b : 0 \}$, $\mathcal{R} = \{ a \rightarrow b \}$ and $\mathcal{L} = \{ f(b), a, b \}$. The language $\mathcal{L}$ is $\mathcal{R}$-closed and regular. There exists no $\mathcal{R}$-closed RDFC tree automaton recognizing $\mathcal{L}$. In any $\mathcal{R}$-closed RDFC tree automaton, $a$ and $b$ need to be recognized by the same state, say $q$, and thus $f(b)$ needs to be recognized using a transition $f(q) \rightarrow q_f$ where $q_f$ is final. Thus, this automaton recognizes $f(a)$ which does not belong to $\mathcal{L}$.

Such approximations are thus out of the scope of Theorem 42, and cannot be found by enumerating $E_C$, because $E$ contains $E_R$ and the completed automata are $\mathcal{R}$-closed. However, the above approximation is in the scope of Theorem 32. We think that it is possible to explore the set of all possible equation sets using $E = E_r \cup E_F$ where $E_F$ is contracting on $\mathcal{T}(\mathcal{F})$ and to prune the search space using Counter Example Guided Abstraction Refinement like [19]. This would permit to have an efficient equation generation for general TRSs and widen its applicability to non-terminating functional programs, cryptographic protocols, etc.

A last perspective is to extend those results to non-left-linear TRSs. Dealing with regular languages and non-left-linear rules is known to be more challenging than the left-linear case [24, 3, 7]. Nevertheless, there could be a nice surprise here. For non-left-linear TRSs, completion is known to be sound and precise as long as the completed tree automaton is kept deterministic [8]. Completion itself does not preserve determinism but, in Section 8, all the completed automata of the experiments are deterministic. This is a consequence of the fact that $E$ contains $E_r$ (makes the automaton $\chi$-deterministic) and $E_R$ (merges all states related by an $\epsilon$-transition). Thus, when using $E = E_R \cup E_r \cup E_C$, completion may build precise over-approximations for non-left-linear TRSs as it does for left-linear ones.

Acknowledgements Many thanks to the anonymous referees for their valuable comments.

References


11 T. Genet. Automata Completion and Regularity Preservation. Technical report, INRIA, 2017. [https://hal.inria.fr/hal-01501744](https://hal.inria.fr/hal-01501744).
## A Experiments

<table>
<thead>
<tr>
<th>Timbuk Spec.</th>
<th>Description</th>
<th>P/C</th>
<th>Comp. Time</th>
<th>Eq. Gen. Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>delete</td>
<td>not (member A (delete A A_and_B_list))</td>
<td>P</td>
<td>0.01s</td>
<td>0.01s</td>
</tr>
<tr>
<td>delete2</td>
<td>(member _ B (delete A A_and_B_list))</td>
<td>P</td>
<td>0.01s</td>
<td>0.01s</td>
</tr>
<tr>
<td>deleteBasic</td>
<td>(delete A A_and_B_list) removes all occurrences of A</td>
<td>P</td>
<td>0.01s</td>
<td>0.01s</td>
</tr>
<tr>
<td>reverseFirstOrder</td>
<td>reverse [A,...A,B,...,B] does not produce lists with a A before a B</td>
<td>P</td>
<td>0.01s</td>
<td>0.03s</td>
</tr>
<tr>
<td>reverseFirstOrder2</td>
<td>invsorted (reverse [A,...A,B,...,B])</td>
<td>P</td>
<td>0.02s</td>
<td>0.13s</td>
</tr>
<tr>
<td>incTree</td>
<td>not (member 0 (increment nat_tree))</td>
<td>P</td>
<td>0.08s</td>
<td>1.05s</td>
</tr>
<tr>
<td>replaceTree</td>
<td>not (member A (replace A C A_and_B_tree))</td>
<td>P</td>
<td>0.44s</td>
<td>6.13s</td>
</tr>
<tr>
<td>orderedTree</td>
<td>ordered ordered_A_and_B_tree</td>
<td>P</td>
<td>0.16s</td>
<td>6.73s</td>
</tr>
<tr>
<td>insertTree</td>
<td>ordered (insert A_and_B_list emptyTree)</td>
<td>P</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>orderedTreeTraversal</td>
<td>sorted (infix-traversal ordered_A_and_B_list)</td>
<td>P</td>
<td>0.13s</td>
<td>1.71s</td>
</tr>
<tr>
<td>orderTreeTraversalBug</td>
<td>sorted (prefix-traversal ordered_A_and_B_list)</td>
<td>C</td>
<td>0.2s</td>
<td>-</td>
</tr>
<tr>
<td>mapPlus</td>
<td>no 0 in (map (plus 1) nat_list)</td>
<td>P</td>
<td>0.02s</td>
<td>0.08s</td>
</tr>
<tr>
<td>filterEven</td>
<td>not (exists even (filter odd nat_list))</td>
<td>P</td>
<td>0.12s</td>
<td>1.16s</td>
</tr>
<tr>
<td>filterEvenBug</td>
<td>not (exists odd (filter odd nat_list))</td>
<td>C</td>
<td>0.09s</td>
<td>-</td>
</tr>
<tr>
<td>insertionSort</td>
<td>(sorted _ leq (sort _ leq A_and_B_list))</td>
<td>P</td>
<td>0.04s</td>
<td>0.11s</td>
</tr>
<tr>
<td>insertionSortBug</td>
<td>(sorted _ geq (sort _ leq A_and_B_list))</td>
<td>C</td>
<td>0.59s</td>
<td>-</td>
</tr>
<tr>
<td>filterNz</td>
<td>(forall nz (filter nz nat_list))</td>
<td>P</td>
<td>0.01s</td>
<td>0.11s</td>
</tr>
<tr>
<td>mapTree</td>
<td>no 0 in (map (plus 1) nat_tree)</td>
<td>P</td>
<td>0.03s</td>
<td>16.15s</td>
</tr>
<tr>
<td>mapTree2</td>
<td>not (member 0 (map (plus 1) nat_tree))</td>
<td>P</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>reverse</td>
<td>(sorted _ geq (reverse ordered_A_and_B_list))</td>
<td>P</td>
<td>0.04s</td>
<td>0.47s</td>
</tr>
<tr>
<td>mapSquare</td>
<td>(filter (eq 2) (map square nat_list))</td>
<td>P</td>
<td>0.31s</td>
<td>4.25s</td>
</tr>
<tr>
<td>foldRightMult</td>
<td>(foldRight mult nonzero_nat_list 1) is not 0</td>
<td>P</td>
<td>0.01s</td>
<td>0.01s</td>
</tr>
<tr>
<td>foldRightMult2</td>
<td>(foldRight mult nonzero_nat_list 3) is not 2</td>
<td>P</td>
<td>0.05s</td>
<td>0.29s</td>
</tr>
<tr>
<td>foldLeftPlus</td>
<td>even (foldLeft plus 0 even_nat_list)</td>
<td>P</td>
<td>0.01s</td>
<td>0.21s</td>
</tr>
</tbody>
</table>

The above table gives a summary of the experiments carried out with Timbuk. The source of the programs, trace of execution, Coq certificates, etc. can be found here: [http://people.irisa.fr/Thomas.Genet/timbuk/funExperiments/](http://people.irisa.fr/Thomas.Genet/timbuk/funExperiments/). The 'Timbuk Spec.' column gives the name of the Timbuk specification file that was used (it is also available in Timbuk's distribution). The first 11 examples are first order programs and the 13 remaining are higher-order programs. The 'Description' column gives a short description of the property.
we want to prove. In the corresponding Timbuk specification this is the initial language and encoded by either a tree automaton or a simplified regular expression [13]. The 'P/C' column says if Timbuk has done a (P)roof of the property or found a (C)ounter example. 'Comp. Time' stands for completion time and 'Eq. Gen. Time' for equation generation time. On some examples, the equation generation algorithm times out and completion cannot be performed.

B Additional proofs

This section contains some the proofs of [11].

| Lemma (16). For all RF tree automata B, for all states q ∈ B there exists a natural number k ∈ N for which the set \([q]^B_k\) is a fixpoint. |

**Proof.** We make a proof by contradiction. Assume that one set of state representatives \([q]^B_k\) is infinite. Let \(Q\) be the set of states of \(B\) and \(t ∈ [q]^B\) be a term s.t. \(|t| > \text{Card}(Q)\). Assume that we label each subterm of \(t\) by the state recognizing it in \(B\). Since height of \(t\) is greater than \(\text{Card}(Q)\), by pigeonhole principle we know that there exists \(q' ∈ B\) and a path in the tree \(t\) such that \(q'\) appears at least two times. Let \(p, r ∈ Pos(t)\) be the positions of the two subterms recognized by \(q'\). By definition of state representatives, we know that \(t|_p ∈ [q']^B\) and \(t|r ∈ [q']^B\). Since \(p\) and \(r\) are on the same path, we know that \(t|_p\) is a strict subterm of \(t|r\). (or the opposite). This contradicts Definition 14 that forbids a term and a strict subterm to belong to the same set of representatives.

| Lemma (19). Let \(B = (F, Q, Q_f, ∆)\) be an RDFC tree automaton and \(E_B = \text{A2E}(B)\). For all \(s ∈ T(F)\), there exists a unique state \(q ∈ Q\) such that \(s →^*_B q\) and for all state representatives \(u ∈ [q]^B\), \(s =_{E_B} u\). |

**Proof.** We make a proof by induction on the height of \(s\). If \(s\) is a constant, since \(B\) is complete and deterministic there exists a unique transition \(s → q ∈ ∆\). By construction of \(E_B\), we know that there are equations with \(s\) on the left-hand side and all state representatives of \([q]^B\) on the right-hand side. For all equations \(s = u\) with \(u ∈ [q]^B\) we thus trivially have \(s =_{E_B} u\). This concludes the base case.

Now, we assume that the property is true for terms of height lesser or equal to \(n\). Let \(s = f(t_1, . . . , t_n)\) where \(t_1, . . . , t_n\) are terms of height lesser or equal to \(n\). Since \(B\) is complete, we know that there exists a state \(q\) such that \(f(t_1, . . . , t_n) →^*_B q\), i.e., there exists states \(q_1, . . . , q_n\) such that \(f(q_1, . . . , q_n) → q ∈ ∆\) and \(t_i →^*_B q_i\) for \(1 ≤ i ≤ n\). Using the induction hypothesis we get that there exist states \(q_i'\) in \(B\) and terms \([q_i']^B\) such that \(t_i →^*_B q_i\) and \(t_i =_{E_B} u_i\) for \(u_i ∈ [q_i']^B\) and for \(1 ≤ i ≤ n\). Since \(B\) is deterministic, from \(t_i →^*_B q_i\) and \(t_i →^*_B q_i'\) we get that \(q_i = q_i'\) and thus \(t_i =_{E_B} u_i\) for \(u_i ∈ [q_i']^B\), with \(1 ≤ i ≤ n\). Besides, since \(f(q_1, . . . , q_n) → q ∈ ∆\), we know that \(E_B\) contains the equations \(f(u_1, . . . , u_n) = u\) for all \(u_i ∈ [q_i']^B\) for all \(1 ≤ i ≤ n\) and for all \(u ∈ [q]^B\). Thus \(q\) is the unique state such that \(f(t_1, . . . , t_n) →^*_B q\). Furthermore, \((f(t_1, . . . , t_n) =_{E_B} f(u_1, . . . , u_n) =_{E_B} u\) for all \(u_i ∈ [q_i']^B\), for all \(1 ≤ i ≤ n\) and for all \(u ∈ [q]^B\).

| Lemma (20). Let \(B = (F, Q, Q_f, ∆)\) be an RDFC tree automaton and \(E_B = \text{A2E}(B)\). For all \(s, t ∈ T(F)\), \(s =_{E_B} t \iff (∃q : \{s, t\} ⊆ L(B, q))\). |

**Proof.** For \(s\) and \(t\), using Lemma 19, we know that there exist unique states \(q, q' ∈ Q\) such that \(s →^*_B q\), \(t →^*_B q'\) and for all state representatives \(u ∈ [q]^B\) and \(v ∈ [q']^B\), we have \(s =_{E_B} u\) and \(t =_{E_B} v\). We first prove the left to right implication. From \(s =_{E_B} t\)
we obtain that \( u = E_B v \), where \( u \) and \( v \) are state representatives. By construction of term representatives, for all states \( q \) we know that \([q]_B\) only contains terms recognized by \( q \) in \( B \). Since \( B \) is deterministic, if \( q \neq q' \) then we can conclude that \([q]_B \cap [q']_B = \emptyset\). Thus, the only possibility to have \( u = E_B v \) is to have an equation \( u = v \) in \( E_B \). This entails that \( u \) and \( v \) belong to the same set of representatives: \([q]_B = [q']_B\), which entails that \( q = q' \). Then \( s \rightarrow q \) and \( t \rightarrow q \) entails that \( \{s,t\} \subseteq L(B,q) \). To prove the right to left implication, it is enough to point out that because of the determinism of \( B \) having \( t \rightarrow q \) (the initial assumption) and having \( t \rightarrow q \) (the fact that \( t \in L(B,q) \)) is possible only if \( q = q' \). This entails that \( u \) and \( v \) have a common set of representatives and thus for all representatives \( u \) of this set \( s = E_B u = E_B t \).

\[ \]

\( \text{Lemma (22).} \) Let \( R \) be a TRS, \( A \) a \( k \)-reduced tree automaton, \( B \) an RDFC tree automaton and \( E_B = A2E(B) \). Let \( A^* \) be the limit of the completion of \( A \) by \( R \) and \( E_B \). For all states \( q \in A^* \), for all terms \( s \in T(F) \) such that \( s \rightarrow^{k^*} q \), there exists a state \( q'_B \in B \), a term \( u \in [q'_B]_B \) such that \( u = E_B s \) and \( u \rightarrow^{k^*} q \).

\textbf{Proof.} Note that if \( A \) is \( k \)-reduced, then so is \( A^* \) (cf. Lemma 44 of [10]). This is easy to figure out since all states added during completion recognize at least one term with \( \rightarrow^{k^*} \), and this is trivially preserved by simplification. By induction on the height of \( s \) we show that the representative \( u \) exists and is recognized by \( q \). If \( s \) is of height 1 (it is a constant) then, by construction of state representatives, we know that \( s \) is a representative. Thus \( s = u \rightarrow^{k^*} q \).

For the inductive case, assume that the property is true for all terms of height lesser or equal to \( n \). Let \( s = f(s_1, \ldots, s_n) \) be a term of height \( n + 1 \). By assumption, we know that \( f(s_1, \ldots, s_n) \rightarrow^{k^*} q \). From \( f(s_1, \ldots, s_n) \rightarrow^{k^*} q \), we obtain that there exists states \( q_1, \ldots, q_n \) of \( A^* \) such that \( s_i \rightarrow^{k^*} q_i \) for \( i = 1, \ldots, n \) and a transition \( f(q_1, \ldots, q_n) \rightarrow q \) in \( A^* \). Using the induction hypothesis on \( q_i \), \( i = 1, \ldots, n \) we get that there exist state representatives \( u_i \) such that \( s_i = E_B u_i \) and \( u_i \rightarrow^{k^*} q_i \) for \( i = 1, \ldots, n \). Then, since \( f(q_1, \ldots, q_n) \rightarrow q \) in \( A^* \) we know that \( f(u_1, \ldots, u_n) \rightarrow^{k^*} q \). If \( f(u_1, \ldots, u_n) \) is a state representative we are done since \( f(s_1, \ldots, s_n) = E_B f(u_1, \ldots, u_n) \) and \( f(u_1, \ldots, u_n) \rightarrow^{k^*} q \). Otherwise, by definition of state representatives, for \( u = f(u_1, \ldots, u_n) \) not to belong to the representatives there is a position \( p \) in \( u \), different from the root position such that the subterm \( u[p] \) is itself a state representative and it belongs to the same class as \( u \), i.e., \( u = E_B u[p] \). Since \( u_1, \ldots, u_n \) are state representatives and \( f(u_1, \ldots, u_n) \) is in the same equivalence class as \( u[p] \), which is a state representative, we know that the equation \( f(u_1, \ldots, u_n) = u[p] \) necessarily belongs to \( E_B \). Besides, for \( u \rightarrow^{k^*} q \) to hold, we know that there exists a state \( q' \) such that \( u[u[p]] = E_B u[q'][p] \rightarrow^{k^*} q \). Thus, \( f(u_1, \ldots, u_n) \rightarrow^{k^*} q \) and \( u[p] \rightarrow^{k^*} q' \). Then, since \( E_B \) contains the equation \( f(u_1, \ldots, u_n) = u[p] \), and since \( A^* \) is simplified w.r.t. \( E_B \), we necessarily have \( q = q' \) in \( A^* \). Finally, we have \( f(s_1, \ldots, s_n) = E_B f(u_1, \ldots, u_n) = E_B u[p] \) and \( u[p] \rightarrow^{k^*} q \) where \( u[p] \) is a state representative.

\[ \]

\( \text{Lemma (21).} \) Let \( B = (F,Q,Q,f,\Delta) \) be an RDFS tree automaton. If \( E_B \) is the set of equations inferred from \( B \) then \( T(F)/=_{E_B} \) is finite.

\textbf{Proof.} Using Lemma 19, we know that for all terms \( t \in T(F) \) there exists a state \( q \in Q \) and a state representative \( u \in [q]_B \) such that \( t \rightarrow q \) and \( t = E_B u \). Since the number of states of \( B \) is finite, and since the set of state representatives \( u \) is finite for all states of \( B \) (Lemma 16), so is the number of equivalence classes of \( T(F)/=_{E_B} \).\[ ]
\textbf{Lemma (22).} Let $\mathcal{R}$ be a TRS, $\mathcal{A}$ a $\kappa$-reduced tree automaton, $\mathcal{B}$ an RDFA tree automaton and $E_B = A2E(\mathcal{B})$. Let $A^*$ be the limit of the completion of $\mathcal{A}$ by $\mathcal{R}$ and $E_B$. For all states $q \in A^*$, for all terms $s \in T(\mathcal{F})$ such that $s \rightarrow_{A^*}^* q$, there exists a state $q_B \in \mathcal{B}$, a term $u \in [q_B]_E$ such that $u = E_B s$ and $u \rightarrow_{A^*}^* q$.

\textbf{Proof.} Note that if $\mathcal{A}$ is $\kappa$-reduced, then so is $A^*$ (cf. Lemma 44 of [10]). This is easy to figure out since all states added during completion recognize at least one term with $\rightarrow_{A^*}^*$, and this is trivially preserved by simplification. By induction on the height of $s$ we show that the representative $u$ exists and is recognized by $q$. If $s$ is of height 1 (it is a constant) then, by construction of state representatives, we know that $s$ is a representative. Thus $u = s \rightarrow_{A^*}^* q$.

For the inductive case, assume that the property is true for all terms of height lesser or equal to $n$. Let $s = f(s_1, \ldots, s_n)$ be a term of height $n + 1$. By assumption, we know that $f(s_1, \ldots, s_n) \rightarrow_{A^*}^* q$. From $f(s_1, \ldots, s_n) \rightarrow_{A^*}^* q$, we obtain that there exists states $q_1, \ldots, q_n$ of $A^*$ such that $s_i \rightarrow_{A^*}^* q_i$ for $i = 1, \ldots, n$ and a transition $f(q_1, \ldots, q_n) \rightarrow q$ in $A^*$. Using the induction hypothesis on $q_i$, $i = 1, \ldots, n$ we get that there exist state representatives $u_i$ such that $s_i = E_B u_i$ and $u_i \rightarrow_{A^*}^* q_i$. Then, since $f(q_1, \ldots, q_n) \rightarrow q$ in $A^*$ we know that $f(u_1, \ldots, u_n) \rightarrow_{A^*}^* q$. If $f(u_1, \ldots, u_n)$ is a state representative we are done since $f(s_1, \ldots, s_n) = E_B f(u_1, \ldots, u_n)$ and $f(u_1, \ldots, u_n) \rightarrow_{A^*}^* q$. Otherwise, by definition of state representatives, for $u = f(u_1, \ldots, u_n)$ not to belong to the representatives there is a position $p$ in $u$, different from the root position such that the subterm $u_p$ is itself a state representative and it belongs to the same class as $u$, i.e., $u = E_B u_p$. Since $u_1, \ldots, u_n$ are state representatives and $f(u_1, \ldots, u_n)$ is in the same equivalence class as $u_p$, which is a state representative, we know that the equation $f(u_1, \ldots, u_n) = u_p$ necessarily belongs to $E_B$. Besides, for $u \rightarrow_{A^*}^* q$ to hold, we know that there exists a state $q'$ such that $u|_{u_p} \rightarrow_{A^*}^* u' \rightarrow_{A^*}^* q$. Thus, $f(u_1, \ldots, u_n) \rightarrow_{A^*}^* q$ and $u_p \rightarrow_{A^*}^* q'$. Then, since $E_B$ contains the equation $f(u_1, \ldots, u_n) = u_p$, and since $A^*$ is simplified w.r.t. $E_B$, we necessarily have $q = q'$ in $A^*$. Finally, we have $f(s_1, \ldots, s_n) = E_B f(u_1, \ldots, u_n) = E_B u_p$ and $u_p \rightarrow_{A^*}^* q$, where $u_p$ is a state representative.

\textbf{Lemma (30).} Let $\mathcal{R}$ be a TRS over $\mathcal{F}$, $S \subseteq T(\mathcal{F})$, and $\mathcal{B}$ an RDFA automaton such that $L(\mathcal{B}) \supseteq R^*(S)$ and $L(\mathcal{B})$ is $\mathcal{R}$-closed. If $E_B = A2E(\mathcal{B})$ then $R_{E_B}(S) \subseteq L(\mathcal{B})$.

\textbf{Proof.} We prove that for all natural number $k \geq 0$, if $s \in S$ and $s \rightarrow_{R/E_B}^k t$ then $t \in L(\mathcal{B})$.

For the inductive case, assume that the property is true for a given $k$ and we show that it is true for $k + 1$. Let $s \rightarrow_{R/E_B}^{k+1} t$, i.e., we have terms $s', s''$, and $t'$ such that $s \rightarrow_{R/E_B}^k s' = E_B s'' \rightarrow_{R/E_B} s'' t' = E_B t$. Using the induction hypothesis, we get that $s'$ is recognized by $\mathcal{B}$. Since $L(\mathcal{B})$ is $\mathcal{R}$-closed, we know that $t'$ is also recognized by $\mathcal{B}$. Thus, there exists a final state $q_f$ such that $t' \rightarrow_{\mathcal{B}}^* q_f$. Finally, as above, applying Lemma 20 on the fact that $t' \rightarrow_{\mathcal{B}}^* q_f$ and $t = E_B t$ gives us that $t \rightarrow_{E_B}^* q_f$. 

\textbf{References}