Checking that Equations on Terms define a Finite Set of Equivalence Classes

Yohan Boichut, Thomas Genet

Checking that Equations on Terms define a Finite Set of Equivalence Classes

Yohan Boichut[a,*], Thomas Genet[b]

[a] *LIFO, UR 4022 – Université d'Orléans, France*

[b] *IRISA, UMR 6074, France*

Abstract

Given a set of equations on terms, we are interested in checking if the set of equivalence classes defined by these equations is finite or not. This theoretical problem finds applications in program verification, where finiteness of the set of equivalence classes ensures termination of the verification. We first show that the problem is undecidable in general by reducing it to a problem in group theory. In a second part, we propose a semi-algorithm for this problem that can either show that the set of equivalence classes is finite or loop. This algorithm relies on the Myhill-Nerode Theorem and tries to build a deterministic finite tree automaton recognizing the equivalence classes defined by the equations.

1. Introduction

Let $\mathcal{F}$ be a ranked alphabet and $\mathcal{X}$ be a set of variables. Let $\mathcal{T}(\mathcal{F}, \mathcal{X})$ be the set of terms built over $\mathcal{F}$ and $\mathcal{X}$, let $\mathcal{T}(\mathcal{F})$ be the set of ground terms (without variables). Let E be a set of equations, i.e., pairs of terms of $\mathcal{T}(\mathcal{F}, \mathcal{X})$. The set E defines an equivalence relation $=_E$. Let $\mathcal{T}(\mathcal{F})/_{=E}$ be the set of *equivalence classes* of $\mathcal{T}(\mathcal{F})$ modulo $=_E$. In this paper, we study *finiteness* of $\mathcal{T}(\mathcal{F})/_{=E}$, i.e., finiteness of the set of equivalence classes.

*Corresponding author

Email addresses: yohan.boichut@univ-orleans.fr (Yohan Boichut), thomas.genet@irisa.fr (Thomas Genet)

Recenlty, checking if $\mathcal{T}(\mathcal{F})/_{=E}$ is finite found applications in the domain of functional program verification [6]. When terms are used to model program states and the program semantics is modelled by a term rewriting systems, equations can be used to define abstractions. When the set of program states is infinite, if E defines an abstraction with a finite set of equivalence classes, then the verification on the *abstracted semantics* can be performed using finite model-checking techniques [5].

In this paper, we first show that deciding if $\mathcal{T}(\mathcal{F})/_{=E}$ is finite is impossible in general. This is a direct consequence of a well-known result of group theory. In a second part, for a given set E, we propose a semi-algorithm to check if $\mathcal{T}(\mathcal{F})/_{=E}$ is finite. The algorithm applies the Myhill-Nerode theorem [3]. The Myhill-Nerode theorem ensures that $\mathcal{T}(\mathcal{F})/_{=E}$ is of finite index (i.e. has a finite set of equivalence classes) iff those equivalence classes are regular (i.e. can be recognized by a tree automaton). Our algorithm relies on this correspondance and tries to build a finite tree automaton recognizing the equivalence classes of $\mathcal{T}(\mathcal{F})/_{=E}$. We also present an implementation of this algorithm and show that it succeeds in showing finiteness of $\mathcal{T}(\mathcal{F})/_{=E}$ on several functional program static analysis problems borrowed from [6].

2. Preliminaries

Comprehensive surveys can be found in [**?** **?**] for term-rewriting systems, and in [3] for tree automata.

Symbols of arity 0 are called constants. We denote by $|t|$ the height of a term $t \in \mathcal{T}(\mathcal{F})$. It is recursively defined as follows:

$$|t| = \begin{cases} \max(|t_1|, \ldots, |t_n|) + 1 & \text{if } t = f(t_1, \ldots, t_n) \text{ and } n > 0 \\ 1 & \text{if } t \text{ is a variable or a constant} \end{cases} \quad (1)$$

A substitution is a function σ from $\mathcal{X}$ into $\mathcal{T}(\mathcal{F}, \mathcal{X})$, which can be uniquely extended to an endomorphism of $\mathcal{T}(\mathcal{F}, \mathcal{X})$ also denoted by σ. We note $\Sigma(\mathcal{T}(\mathcal{F}, \mathcal{X}), \mathcal{X})$ is the set of all substitutions from $\mathcal{X}$ to $\mathcal{T}(\mathcal{F}, \mathcal{X})$. A *set of equations* E is a set of pairs of the form $l = r$ where $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. A position p for a term t is a word over $\mathbb{N}$. The empty sequence ϵ denotes the top-most position. The set $\mathcal{P}os(t)$ of positions of a term t is inductively defined by: (a) $\mathcal{P}os(t) = \{\epsilon\}$ if $t \in \mathcal{X}$, (b) $\mathcal{P}os(f(t_1, \ldots, t_n)) = \{\epsilon\} \cup \{i.p \mid 1 \leq i \leq n \wedge$

$p \in \mathcal{P}os(t_i)\}$. If $p \in \mathcal{P}os(t)$, then $t|_p$ denotes the subterm of t at position p and $t[s]_p$ denotes the term obtained by replacement of the subterm $t|_p$ at position p by the term s.

Definition 1 (E-equivalence)**.** *For two ground terms $t, t' \in \mathcal{T}(\mathcal{F})$ and an equation $e : l = r$, we say that $t =_e t'$ if there exists a substitution $\tau : \mathcal{X} \mapsto \mathcal{T}(\mathcal{F})$ such that $l\tau = t$ and $r\tau = t'$. The equivalence relation $=_E \subseteq \mathcal{T}(\mathcal{F}) \times \mathcal{T}(\mathcal{F})$ is the smallest congruence containing the relation $\{(t, t') \in \mathcal{T}(\mathcal{F}) \times \mathcal{T}(\mathcal{F}) \mid \exists e \in E.\ t =_e t'\}$.* $\diamond$

For $t \in \mathcal{T}(\mathcal{F})$, we write $[t]_E$ for the equivalence class of t, i.e. $[t]_E = \{u \in \mathcal{T}(\mathcal{F}) \mid u =_E t\}$ and $\mathcal{T}(\mathcal{F})/_{=E} = \{[t]_E \mid t \in \mathcal{T}(\mathcal{F})\}$. We now define tree automata. Let Q be a finite set of symbols with arity 0, called *states*, such that $Q \cap \mathcal{F} = \emptyset$.

Definition 2 (Normalised transition)**.** *Let $f \in \mathcal{F}$ be a symbol of arity n. A* normalised transition *is a pair $f(q_1, \ldots, q_n) \to q$, where $q_1, \ldots, q_n, q \in Q$.* $\diamond$

Definition 3 (Tree automaton)**.** *A (bottom-up, finite) tree automaton, simply called a tree automaton or an automaton in the sequel, is a tuple $A = \langle \mathcal{F}, Q, \Delta \rangle$, where Δ is a set of normalised transitions.* $\diamond$

The set of transitions Δ defines a *rewriting relation* $\to_\Delta$ on $\mathcal{T}(\mathcal{F} \cup Q)$ defined as follows. Let $c \to q$ be a normalised transition of Δ and $t \in \mathcal{T}(\mathcal{F} \cup Q)$. If there exists a position $p \in \mathcal{P}os(t)$ such that $t|_p = c$ then $t \to_\Delta t[q]_p$. We write $\to_\Delta^*$ for the reflexive transitive closure of $\to_\Delta$. For a tree automaton $A = \langle \mathcal{F}, Q, \Delta \rangle$, we often write $\to_A$ and $\to_A^*$ instead of $\to_\Delta$ and $\to_\Delta^*$. In the same way, we use $c \to q \in A$ and $q \in A$ as shorthands for $c \to q \in \Delta$ and $q \in Q$. An automaton A is *deterministic* if for all $t \in \mathcal{T}(\mathcal{F})$ there is at most one state q such $t \to_A q$. We denote by $\Sigma(Q, \mathcal{X})$ the set of substitutions replacing variables by states in terms.

Definition 4 (Recognized language)**.** *The tree language recognized by A in a state q is $\mathcal{L}(A, q) = \{t \in \mathcal{T}(\mathcal{F}) \mid t \to_A^* q\}$.* $\diamond$

Example 1 (Tree automaton, recognized language, substitutions)**.** *Let $\mathcal{F} = \{f, a, b\}$ and $A = \langle \mathcal{F}, Q, \Delta \rangle$, with $Q = \{q_1, q_2\}$, and $\Delta = \{f(q_1) \to q_1, a \to q_1, b \to q_2, f(q_2) \to q_1\}$. The languages recognized by q_1 and q_2 are: $\mathcal{L}(A, q_1) = \{f^i(a) \mid i \geq 0\} \cup \{f^i(b) \mid i > 0\}$ and $\mathcal{L}(A, q_2) = \{b\}$. The automaton A is deterministic. The substitution $\sigma = \{x \mapsto q_1, y \mapsto q_2\}$ belongs to $\Sigma(Q, \mathcal{X})$ and $f(x, y)\sigma = f(q_1, q_2)$.*

We also introduce the notion of a complete automaton. A complete automaton recognizes any term of $\mathcal{T}(\mathcal{F})$ in a state.

Definition 5. *Let A be a tree automaton. Given an alphabet $\mathcal{F}$, A is complete iff for any term $t \in \mathcal{T}(\mathcal{F})$ there exists a state of A such that $t \to_A^* q$.* $\diamond$

3. Checking if $\mathcal{T}(\mathcal{F})/_{=E}$ is finite is undecidable in general

When E is a set of *ground equations*, checking if $\mathcal{T}(\mathcal{F})/_{=E}$ is finite is *decidable*. Using a congruence closure algorithm [1], it is possible to build a terminating and confluent term rewriting system $\mathcal{R}$ for deciding $=_E$. Then, we can compute the tree automaton recognizing the set of normal forms of $\mathcal{R}$ [2] and check that this automaton recognizes a finite set of terms. More generally, as soon as there exists a left-linear, confluent, terminating term rewriting system deciding $=_E$, we can verify that its set of normal forms is finite to ensure that $\mathcal{T}(\mathcal{F})/_{=E}$ is finite. It is also decidable when the set E enjoys strong syntactic restrictions [4].

However, checking if $\mathcal{T}(\mathcal{F})/_{=E}$ is finite is undecidable in general. In [10], F. Otto proves a result on groups which is close to ours. In the following, we simply recall the sketch of the proof of [10] and show how to relate is to the undecidability of finiteness of $\mathcal{T}(\mathcal{F})/_{=E}$. In group theory, some properties of groups are known as Markov properties. For instance, the property of being a finite group, i.e., having a finite underlying set is a Markov property. In [9], Theorem 4.1. recalls that for any Markov property P, there exists no algorithm to decide whether a finitely presented group have the property P.

What remains to be done is to relate our problem on $\mathcal{T}(\mathcal{F})/_{=E}$ to groups. First, let us consider the same problem on words instead of trees. Words are built using letters from a set of symbols Ω, the (associative) concatenation symbol '.' and the empty word λ (the neutral element). Let us consider the case where Ω is a set of letters $\{a_1, a_1^{-1}, \ldots, a_n, a_n^{-1}\}$ and E is a set of equations $u = v$ where $u, v \in \Omega^*$. The set E is the presentation of the group. Furthermore, we need another set of equations E_{inv} encoding the behavior of a letter w.r.t. its inverse $E_{inv} = \{a_i.a_i^{-1} = \lambda,\ a_i^{-1}.a_i = \lambda \mid i = 1 \ldots n\}$. Any group can be encoded in such a Ω, E and E_{inv}. If the group is finitely presented then so are Ω, E and E_{inv}. Thus, if there exists an algorithm to check if $\Omega^*/_{=E \cup E_{inv}}$ is finite then we have an algorithm to check if the group is finite, which contradicts the above theorem from [9].

This undecidability result can be lifted to terms by encoding words into terms by defining in $\mathcal{F}$ a unary function symbol for each letter of Ω and a constant symbol for λ, i.e., we encode any word of the form $a_1.a_2.\lambda$ by the term $a_1(a_2(\lambda))$. All equations on words can be transformed into equations on terms, e.g., the set of equations E_{inv} becomes $E_{inv} = \{a_i(a_i^{-1}(x)) = x,\ a_i^{-1}(a_i(x)) = x \mid i = 1 \dots n\}$. Hence, $\mathcal{T}(\mathcal{F})/_{=E \cup E_{inv}}$ is also a finitely presented group whose finiteness cannot be decided.

4. Representing equivalence relations by tree automata

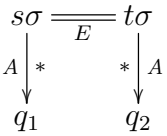
As shown above, checking if $\mathcal{T}(\mathcal{F})/_{=E}$ is finite is undecidable in general. Now, we propose a *criterion* for checking if $\mathcal{T}(\mathcal{F})/_{=E}$ is finite for a given set of equations E. This criterion applies the Myhill-Nerode theorem which ensures that $\mathcal{T}(\mathcal{F})/_{=E}$ has a finite set of equivalence classes iff those equivalence classes can be recognized by a tree automaton [3]. Given E, our criterion incrementally builds a tree automaton recognizing all E-equivalent terms. Using tree automata to represent E-equivalence has already been done in [7], where tree automata represent terms reachable by rewriting modulo equations. For such an automaton A, if two terms $t, t' \in \mathcal{T}(\mathcal{F})$ belong to the same language, i.e., $t, t' \in \mathcal{L}(A, q)$, then t and t' are E-equivalent. We can take advantage of several of their definitions and theorems. A first notion we can use is the simplification of a tree automaton w.r.t. a set of equations. Simplification uses equations to merge states recognizing E-equivalent terms in a tree automaton. Roughly, the merging of two states q_1 and q_2 is performed by renaming every occurrence of q_2 by q_1 in the sets of states and in the set of transitions of the automaton.

Definition 6 (Renaming states in tree automata [7])**.** *Let $A = \langle \mathcal{F}, Q, \Delta \rangle$ be a tree automaton, and α a function $\alpha : Q \mapsto Q$. We denote by $A\alpha$ the tree automaton where every occurrence of q is replaced by $\alpha(q)$ in Q, and in every left and right-hand side of every transition of Δ.* ◇

When $\alpha = \{q_a \mapsto q_b\}$, $A' = A\alpha$ is the automaton where every occurrence of q_a has been replaced by q_b.

Definition 7 (Simplification relation [7])**.** *Let $A = \langle \mathcal{F}, Q, \Delta \rangle$ be a tree automaton and E be a set of equations. The simplification relation, denoted by $\rightsquigarrow_E$, is defined as follows: $A \rightsquigarrow_E A'$ if there exist distinct states $q_1, q_2 \in Q$,*

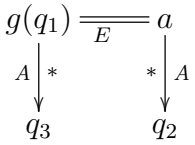
an equation $s = t \in E$, and a substitution $\sigma \in \Sigma(Q, \mathcal{X})$, such that $s\sigma \to_A^* q_1$, $t\sigma \to_A^* q_2$, and $A' = A\{q_1 \mapsto q_2\}$.



◇

Example 2. *Let A be a tree automaton such that $\Delta = \{f(q_1, q_2) \to q_0, a \to q_1, a \to q_2, g(q_1) \to q_3\}$.*

- *If $E = \{g(x) = a\}$, we have $\sigma = \{x \mapsto q_1\}$ and*



  Hence, $A \rightsquigarrow_E A'$ where $A' = A\{q_3 \mapsto q_2\}$. Note that, if we choose $A'' = \{q_2 \mapsto q_3\}$ then we obtain an automaton that is equivalent to A' modulo a bijective renaming of states.

- *If $E = \{f(x, x) = g(x)\}$ then there is no substitution σ such that $f(x, x)\sigma = f(q_1, q_2)$: the automaton is unchanged, and a is still recognized in two distinct states q_1 and q_2.*

Note that the $\rightsquigarrow_E$ is not a deterministic relation, i.e, we may have a choice on the equation to use and on the renaming to apply. Fortunately, Lemma 17 and Lemma 18 of [7] show, respectively, that the relation $\rightsquigarrow_E$ is well-founded and that it is locally confluent modulo renaming of states. This implies that the normal forms of $\rightsquigarrow_E$ are unique modulo renaming of states. Thus, if A, A_1 and A_2 are automata such that $A \rightsquigarrow_E^* A_1$, $A \rightsquigarrow_E^* A_2$ and A_1 and A_2 are in normal form w.r.t. $\rightsquigarrow_E$ then A_1 and A_2 are equal modulo a bijective renaming of their states. Since this normal form is unique (modulo renaming) we can define $NF_E(A)$, the complete simplification of A w.r.t. E.

Definition 8 ($NF_E(A)$)**.** *Let $A = \langle \mathcal{F}, Q, \Delta \rangle$ be a tree automaton and E be a set of equations. The automaton $NF_E(A)$ is the automaton A' such that $A \rightsquigarrow_E^* A'$ and A' is in normal form w.r.t. $\rightsquigarrow_E$.* ◇

In [7], every constructed automaton is R/E-coherent, meaning that it closely represent the rewriting relation of terms w.r.t. a term rewriting system $\mathcal{R}$ and a set of equations E. In our setting, $\mathcal{R}$ is empty so our definition is a simple restriction to the pure equational case.

Definition 9 (E-coherent automaton)**.** *Let $A = \langle \mathcal{F}, Q, \Delta \rangle$ be a tree automaton and E a set of equations. The automaton A is said to be* E-coherent *if for all states $q \in Q$, there exists a term $s \in \mathcal{T}(\mathcal{F})$ such that $s \to_A^* q$, and for all $t \in \mathcal{T}(\mathcal{F})$ s.t. $t \to_A^* q$ then $s =_E t$.* ◇

Example 3. *Any automaton where all terms are recognized in distinct states is E-coherent, for any equation system E. For example, the automaton whose states are $Q = \{q_1, q_2\}$ and transitions are $a \to q_1, b \to q_2$, is so. This observation is important, because in the following we build such automata and we need E-coherence to hold on them. On the other hand, the same automaton enriched with either (*i*) the transition $b \to q_1$, or (*ii*) the state q_3, is not E-coherent if $E = \emptyset$. In the case (*i*) this is because a, b are both recognized in q_1 and are not equal* modulo E*, and in the case (*ii*) because q_3 does not recognize any term. The automaton (*i*) becomes E-coherent with $E = \{a = b\}$.*

A nice property is that the simplification relation $\rightsquigarrow_E$ preserves R/E-coherence. This can easily be exploited to prove E-coherence preservation on $NF_E(A)$.

Proposition 1. *Let A be a tree automaton and E a set of equations. If A is E-coherent then so is $NF_E(A)$.*

Proof. In [7], automata have normalized transitions and ϵ-transitions, i.e., transitions of the form $q \to q'$ where $q, q' \in Q$. Roughly, normalized transition recognize E-equivalent terms and ϵ-transitions represent rewriting steps between them. In their setting, an automaton A is $\mathcal{R}/E$-coherent if for all states $q \in Q$, there exists a term $s \in \mathcal{T}(\mathcal{F})$ such that $s \to_A^* q$ (using no ϵ-transition), and for all $t \in \mathcal{T}(\mathcal{F})$:

1. if $t \to_A^* q$ (using no ϵ-transitions), then $s =_E t$, and
2. it $t \to_A^* q$ (possibly using ϵ-transitions), then $s \to_{\mathcal{R}/E}^* t$, where $\to_{\mathcal{R}/E}$ is the rewriting relation with a term rewriting system $\mathcal{R}$ modulo a set of equations E.

In our tree automata there is no ϵ-transition, so E-coherence of A directly implies $\mathcal{R}/E$-coherence of A with an empty term rewriting system $\mathcal{R}$. Since every simplification step with $\rightsquigarrow_E$ preserves $\mathcal{R}/E$-coherence (Theorem 30 of [7]) then $NF_E(A)$ is $\mathcal{R}/E$-coherent. Since A has no ϵ-transitions and $\rightsquigarrow_E$ does not add ϵ-transitions then $NF_E(A)$ has no ϵ-transitions. Thus, $\mathcal{R}/E$-coherence of $NF_E(A)$ implies E-coherence of $NF_E(A)$. □

With this result, we are close to representing the equivalence classes of E using the automaton $NF_E(A)$. However, as it stands, the automaton $NF_E(A)$ may still have more states than equivalence classes. We illustrate this with the following example.

Example 4. *Let A be the automaton with transitions $a \to q_a$, $f(q_a) \to q_1$, $b \to q_b$, $f(q_b) \to q_2$, where q_1 recognizes $f(a)$ and q_2 recognizes $f(b)$. Let $E = \{a = b\}$. In the automaton $NF_E(A)$, states q_a and q_b are merged. Assume that in $NF_E(A)$ we choose to replace each occurrence of q_b by q_a, then $NF_E(A)$ has transitions $a \to q_a$, $b \to q_a$, $f(q_a) \to q_1$ and $f(q_a) \to q_2$. Note that $NF_E(A)$ is E-coherent. However, we have the following two derivations: $f(a) \to_A f(q_a) \to_A q_1$ and $f(b) \to_A f(q_a) \to_A q_2$. Thus, terms $f(a)$ and $f(b)$ that belong to the same equivalence class (because $f(a) =_E f(b)$), are recognized in two different states. The term $f(a)$ can also be recognized into states q_1 and q_2. The automaton $NF_E(A)$ is not deterministic though A is deterministic.*

Thus, automaton simplification may not preserve determinism and may result in a tree automaton with more states than equivalence classes. A simple solution to this problem is to complement the set of equations E with a set E_r of *reflexivity equations*. The set of reflexivity equations is $E_r = \{f(x_1, \ldots, x_n) = f(x_1, \ldots, x_n) \mid f \in \mathcal{F}^n\}$. Simplification of an automaton with $E \cup E_r$ yields an automaton that is deterministic.

Lemma 1 (Automaton $NF_{E \cup E_r}(A)$ is deterministic)**.** *Let A be an automaton, E a set of equations and E_r a set of reflexivity equations. The automaton $NF_{E \cup E_r}(A)$ is deterministic.*

Proof. We make a proof by contradiction. If $NF_{E \cup E_r}(A)$ is not deterministic then there exists two transitions $f(q_1, \ldots, q_n) \to q$ and $f(q_1, \ldots, q_n) \to q'$ in $NF_{E \cup E_r}(A)$. The equation $f(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$ belongs to E_r and can be applied on those two transitions, yielding the merging of states

q and q'. This contradicts the fact that $NF_{E\cup E_r}(A)$ is in normal form w.r.t. $\rightsquigarrow_{E\cup E_r}$. □

Besides, note that the equivalence relation $=_E$ is identical to $=_{E\cup E_r}$. This permits to extend Proposition 1 for $NF_{E\cup E_r}$.

Proposition 2. *Let A be a tree automaton and E a set of equations. If A is E-coherent then so is $NF_{E\cup E_r}(A)$.*

Proof. Since the relation $=_E$ is identical to $=_{E\cup E_r}$, E-coherence and $E\cup E_r$-coherence are also the same. Thus, from E-coherence of A we get $E\cup E_r$-coherence of A. Using Proposition 1, we get that $NF_{E\cup E_r}(A)$ is $E\cup E_r$-coherent, which is equivalent to being E-coherent. Thus, $NF_{E\cup E_r}(A)$ is E-coherent. □

Thus, we can safely use simplification with $NF_{E\cup E_r}$ instead of NF_E.

Definition 10 (The simplified automaton $\mathcal{S}_E(A)$)**.** *Let A be an automaton, E a set of equations and E_r a set of reflexivity equations. We denote by $\mathcal{S}_E(A)$ the automaton $NF_{E\cup E_r}(A)$.* ◇

We obtain the following proposition for $\mathcal{S}_E(A) = NF_{E\cup E_r}(A)$ that combines Lemma 1 and Proposition 2.

Proposition 3. *Let A be an automaton and E a set of equations, $\mathcal{S}_E(A)$ is deterministic and if A is E-coherent then so is $\mathcal{S}_E(A)$.*

5. The criterion for proving that $\mathcal{T}(\mathcal{F})/_{=E}$ is finite

Our semi-algorithm can be summarized as follows: we build a sequence of automata $A_0, A_1, \ldots$ such that A_0 is the empty automaton and, for $i > 0$, A_i contains the transitions of A_{i-1} plus the transitions necessary to recognize all the terms of height i. Each automaton A_i is simplified using $\mathcal{S}_E$. If a tree automaton A_i recognizes all terms of height $i+1$ then it is a fixpoint, i.e. $A_{i+1} = A_i$, and each state of A_i recognizes an equivalence classe of $\mathcal{T}(\mathcal{F})/_{=E}$. Thus $\mathcal{T}(\mathcal{F})/_{=E}$ is finite.

Throughout this section, we denote by T^i the set of terms of $\mathcal{T}(\mathcal{F})$ whose depth is exactly i. To simplify the upcoming definitions, we define a union operator between a tree automaton and a set of transitions.

Definition 11. *Let $A = \langle \mathcal{F}, Q, \Delta \rangle$ an automaton and Δ' a set of transitions on $\mathcal{F}$. We denote by $A \cup \Delta'$ the automaton $\langle \mathcal{F}, Q \cup Q', \Delta \cup \Delta' \rangle$ where $Q = \bigcup_{f(q_1,\ldots,q_n)\to q \in \Delta}(\{q_1, \ldots, q_n, q\})$.* $\diamond$

The following definition describes how to generate a deterministic set of transitions from a term $t \in \mathcal{T}(\mathcal{F})$, such that this set recognizes t in a unique state q_t.

Definition 12 ($Norm$)**.** *Let t be a term of $\mathcal{T}(\mathcal{F})$.*

We define $Norm(t)$ as the set of transitions as follows:

$$Norm(t) = \begin{cases} \{t \to q_t\} & \text{if } t \text{ is a constant} \\ \{f(q_{t_1}, \ldots, q_{t_n}) \to q_t\} \cup \bigcup_{i=1}^{n} Norm(t_i) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

$\diamond$

Example 5. *Let $\mathcal{F} = \{g, f, a\}$ and $t = g(f(a), a)$. Thus, $Norm(t) = \{a \to q_a, f(q_a) \to q_{f(a)}, g(q_{f(a)}, q_a) \to q_{g(f(a),a)}\}$ and $g(f(a), a) \to^*_{Norm(t)} q_{g(f(a),a)}$*

Given $\mathcal{F}$ and E, we now define how to compute a sequence of tree automata $(A_i)^{i\geq 0}$ recognizing the equivalence classes of terms of height lesser or equal to i. In the following, we call this sequence an EC-sequence. Let $A_\emptyset$ be the automaton having an empty set of states and transitions.

Definition 13 (EC-sequence $(A_i)^{i\geq 0}$)**.** *Let $\mathcal{F}$ be an alphabet and E be a set of equations. The EC-sequence for $\mathcal{F}$ and E is the sequence of automata $(A_0, A_1, \ldots)$ such that $A_0 = A_\emptyset$ and, for all $i \geq 0$:*

$$\Delta_{i+1} = \bigcup_{t \in T_{i+1}}(Norm(t))) \qquad \text{and} \qquad A_{i+1} = \mathcal{S}_E(A_i \cup \Delta_{i+1}))$$

Example 6. *Let $\mathcal{F} = \{g, f, a\}$ and $E = \{f(x) = x\}$. We have $A_0 = A_\emptyset$. Then, from $T^1 = \{a\}$, it follows that $\Delta_1 = \{a \to q_a\}$. Note that the simplification step by $\mathcal{S}_E$ has no effect in this case, since the equation of E is not applicable. We then have $T^2 = \{f(a), g(a, a)\}$. After the normalization step, we obtain the set $\Delta_2 = \{a \to q_a,\ f(q_a) \to q_{f(a)},\ g(q_a, q_a) \to q_{g(a,a)}\}$. Next, the simplification step with $\mathcal{S}_E$ and equation $f(x) = x$ merges q_a and $q_{f(a)}$. As a result, we obtain the automaton A_2 whose set of transitions is $\{a \to q_a,\ f(q_a) \to q_a,\ g(q_a, q_a) \to q_{g(a,a)}\}$. Note that $A_0 \neq A_1$, $A_1 \neq A_2$, etc. For this particular $\mathcal{F}$ and E, the sequence will not reach a fixpoint.*

6. Soundness theorem

In this section, we show a soundness theorem for the EC-sequence $(A_i)^{i\geq 0}$: if the computation of the EC-sequence $(A_i)^{i\geq 0}$ reaches a fixpoint, then the set $\mathcal{T}(\mathcal{F})/_{=E}$ has a finite number of equivalence classes. The proof relies on two key properties. The first one states that, for all $i \geq 0$, the automaton A_i is E-coherent. The second property states that if the EC-sequence reaches a fixpoint A_k then this automaton is complete.

Proposition 4. *Given a EC-sequence $(A_i)^{i\geq 0}$, for all $i \geq 0$, A_i is $E-$coherent.*

Proof. Let us proceed by induction on i.

- $A_0 = A_\emptyset$ is the empty automaton which is trivially $E-$coherent.
- We suppose that A_i is an $E-$coherent automaton. We want to prove that the automaton $A_{i+1} = \mathcal{S}_E(A_i \cup \Delta_{i+1})$ is E-coherent. From Proposition 3, we know that $\mathcal{S}_E$ preserves E-coherence. Thus we only need to prove that $A_i \cup \Delta_{i+1}$ is E-coherent. Recall that $\Delta_{i+1} = \bigcup_{t\in T^{i+1}}(Norm(t))$. For any t, $Norm(t)$ ensures that for any $p \in \mathcal{P}os(t)$, $t|_p \rightarrow^*_{Norm(t)} q_{t|_p}$. All states reductions with $Norm(t)$ alone are E-coherent by construction: each state q_t recognizes a single term: t. Thus reductions with Δ_{i+1} are E-coherent. What remains to be shown is that reductions with $A_i \cup \Delta_{i+1}$ are also E-coherent. There are two cases:
  - No states of Δ_{i+1} occurs in A_i then we trivially get that $A_i \cup \Delta_{i+1}$ is E-coherent.
  - Assume that there exists a common state q_t between A_i and Δ_{i+1}. If q_t appears in A_i, this means that it has been added by the normalization of the term t in a tree automaton A_j with $j \leq i$. Since merging steps only adds possible reductions, we necessarily still have $t \rightarrow^*_{A_i} q_t$ in A_i. From E-coherence of A_i we know that, for all terms $u \rightarrow^*_{A_i} q_t$ we have $t =_E u$. Besides, for q_t to appear in Δ_{i+1}, we know that the *unique* term that it can recognize is t. Thus, $A_i \cup \Delta_{i+1}$ is E-coherent.

□

We say that a EC-sequence $(A_i)^{i\geq 0}$ reaches a fixpoint if there exists an index $n \geq 0$ such that $A_n = A_{n+1}$. From the previous property, we know that A_n is E-coherent. The following proposition shows that, additionnally, the automaton A_n is complete, i.e., that every term of $\mathcal{T}(\mathcal{F})$ is recognized by at least one of its state.

Proposition 5. *Let $(A_i)^{i\geq 0}$ be an EC-sequence. For $n \geq 0$, if $A_n = A_{n+1}$ then A_n is complete.*

Proof. For all terms t of $\mathcal{T}(\mathcal{F})$, we show by induction on $k = |t|$ that there exists a state $q \in A_n$ such that $t \to^*_{A_n} q$.

1. The base case is when $|t| = k \leq n$. Then, by construction of A_n, we trivially get that there exists a state q of A_n such that $t \to^*_{A_n} q$.

2. Now we assume that the property is true for all terms t s.t. $|t| \leq k$. Let us prove that this is also true for $t = f(t_1, \ldots, t_m)$ s.t. $|t| = k + 1$. We can use the induction hypothesis on all t_i and we get that there exists $q_1, \ldots, q_m$ such that $t_i \to^*_{A_n} q_i$ for $i = 1 \ldots m$. Thus, $f(t_1, \ldots, t_m) \to^*_{A_n} f(q_1, \ldots, q_m)$. Besides, since $q_i \in A_n$ for $i = 1 \ldots m$, according to Definition 13, we can deduce that there exists $t'_1, \ldots, t'_m$ such that $|t_i| \leq n$ and $t'_i \to^*_{A_n} q_i$. Consequently, $|f(t'_1, \ldots, t'_m)| \leq n + 1$ and, by Definition 13, we know that $f(t'_1, \ldots, t'_m)$ is necessarily recognized by A_{n+1}. Since $A_{n+1} = A_n$, this term is also recognized by A_n, i.e., there exists a state q such that $f(t'_1, \ldots, t'_m) \to^*_{A_n} q$. This implies that there exists states $q'_1, \ldots, q'_m$ and a transition $f(q'_1, \ldots, q'_m) \to q$ such that $t'_i \to^*_{A_n} q'_i$ for $i = 1 \ldots m$. Above, we had that $t'_i \to^*_{A_n} q_i$ and, now, that $t'_i \to^*_{A_n} q'_i$, for $i = 1 \ldots m$. Since the automaton A_n is deterministic we get that $q'_i = q_i$ and thus $f(t'_1, \ldots, t'_m) \to^*_{A_n} f(q_1, \ldots, q_m) \to q$ and finally $f(t_1, \ldots, t_m) \to^*_{A_n} f(q_1, \ldots, q_m) \to q$. This concludes the proof.

□

Now, using the two previous properties, we can state and prove that the existence of a fixpoint in the EC-sequence implies that $\mathcal{T}(\mathcal{F})/_{=E}$ is finite.

Theorem 1. *Given an alphabet $\mathcal{F}$ and a set of equations E, if there exists a EC-sequence $(A_i)^{i\geq 0}$ that reaches a fixpoint, then $\mathcal{T}(\mathcal{F})/_{=E}$ is finite.*

Proof. We make a proof by contradiction. Assume that A_n is the fixpoint and that $\mathcal{T}(\mathcal{F})/_{=E}$ is not finite, i.e., there exists an infinite sequence of terms $t_1, t_2, \ldots$ all belonging to distinct equivalence classes. Since A_n is complete (Proposition 5), there exists states $q_1, q_2, \ldots$ recognizing $t_1, t_2, \ldots$ Since A_n is finite, we know that there exists i and j s.t. $q_i = q_j$. Thus, $t_i \rightarrow^*_{A_n} q_i$ and $t_j \rightarrow^*_{A_n} q_j$ with $q_i = q_j$. Since A_n is E-coherent (Proposition 4), we get that $t_i =_E t_j$, which is a contradiction.

□

7. Relative completeness theorem

Since the problem is undecidable, our criterion cannot be complete. In this section, we prove a *relative* completeness theorem, i.e., if $\mathcal{T}(\mathcal{F})/_{=E}$ is finite then the EC-sequence $(A_i)^{i\geq 0}$ always reaches a fixpoint. We also show that, in the fixpoint automaton, each state recognizes an equivalence class of $\mathcal{T}(\mathcal{F})/_{=E}$. On the contrary, if $\mathcal{T}(\mathcal{F})/_{=E}$ is not finite then the criterion loops infinitely. To prove this relative completeness theorem, we need the notion of a E-saturated automaton, i.e., an automaton that exactly represent the $=_E$ relation. We also show that complete and simplified tree automata are necessarily E-saturated.

Definition 14 (E-saturated automaton)**.** *Let $A = \langle \mathcal{F}, Q, \Delta \rangle$ be a tree automaton and E a set of equations. The automaton A is E-saturated if for all states $q_1 \in Q$ and $q_2 \in Q$ and all terms $s, t \in \mathcal{T}(\mathcal{F})$ such that $s \rightarrow^*_A q_1$, $t \rightarrow^*_A q_2$ and $s =_E t$ then $q_1 = q_2$.* ◇

Lemma 2. *Let A be a tree automaton and E a set of equations. If $\mathcal{S}_E(A)$ is complete then it is E-saturated.*

Proof. We want to prove that for all states $q_1 \in Q$ and $q_2 \in Q$ and all terms $s, t \in \mathcal{T}(\mathcal{F})$ such that $s \rightarrow^*_{\mathcal{S}_E(A)} q_1$, $t \rightarrow^*_{\mathcal{S}_E(A)} q_2$ and $s =_E t$ then $q_1 = q_2$. We make a proof by induction on the number k of equations of E used to deduce $s =_E t$.

- If $k = 0$ this means that s is syntactically equal to t. By proposition 3, we know that $\mathcal{S}_E(A)$ is deterministic thus the term s is necessarily recognized by a single state and $q_1 = q_2$.

- If $k = 1$ this means that $s =_E t$ because of the application of a single equation. Thus, there exists a context $C[\ ]$, an equation $l = r$ and a substitution σ such that $s = C[l\sigma]$ and $t = C[r\sigma]$. Since $s \to^*_{\mathcal{S}_E(A)} q_1$, we know that $C[l\sigma] \to_{\mathcal{S}_E(A)} q_1$ and, thus, that there exists a state q_1' such that $l\sigma \to^*_{\mathcal{S}_E(A)} q_1'$ and $C[l\sigma] \to_{\mathcal{S}_E(A)} C[q_1'] \to_{\mathcal{S}_E(A)} q_1$. Let us assume that l is of the form $l[x_1, \ldots, x_n]$ where $(x_1, \ldots, n_n)$ is the vector of variables occurring in l. Since l may be non-linear, some variables may occur at several positions in this vector. We denote by t_i the terms obtained by applying σ to x_i, i.e. $t_i = x_i\sigma$ for $i = 1 \ldots n$. Since $l\sigma \to^*_{\mathcal{S}_E(A)} q_1'$, we know that there exists states q_i'' such that $t_i \to^*_{\mathcal{S}_E(A)} q_i''$ for all $i = 1 \ldots n$. Let ρ_1 be the substitution $\{x_i \mapsto q_i'' \mid i = 1 \ldots n\}$. Note that, even if l is non-linear, there is a unique state q_i'' to associate to x_i. This is due to the fact that the automaton $\mathcal{S}_E(A)$ is deterministic (Proposition 3). In particular, each term t_i is recognized by a *unique* state q_i''. Thus, we have $l\sigma = l[x_1\sigma, \ldots, x_n\sigma] = l[t_1, \ldots, t_n] \to^*_{\mathcal{S}_E(A)} l[x_1\rho_1, \ldots, x_n\rho_1] \to^*_{\mathcal{S}_E(A)} q_1'$.

  We can make a similar reasoning on $C[r\sigma] \to^*_{\mathcal{S}_E(A)} q_2$ and get that there exists a state q_2' such that $r\sigma \to^*_{\mathcal{S}_E(A)} q_2'$ and $C[r\sigma] \to_{\mathcal{S}_E(A)} C[q_2'] \to_{\mathcal{S}_E(A)} q_2$. As above, we can build a substitution ρ_2 such that $r\sigma = r[x_1\sigma, \ldots, x_m\sigma] = r[t_1, \ldots, t_m] \to^*_{\mathcal{S}_E(A)} r[x_1\rho_2, \ldots, x_m\rho_2] \to^*_{\mathcal{S}_E(A)} q_2$. Furthermore, ρ_1 and ρ_2 agree on each common variables since they necessarily map a common variable x to the *unique* state recognizing $x\sigma$. Let us call ρ the union of substitutions ρ_1 and ρ_2. Finally, since $l = r$ is an equation of E, $l\rho \to^*_{\mathcal{S}_E(A)} q_1'$ and $r\rho \to^*_{\mathcal{S}_E(A)} q_2'$ and $\mathcal{S}_E(A)$ is in normal form w.r.t. $\rightsquigarrow_E$ we know that this equation has necessarily been used in a simplification step and thus $q_1' = q_2'$. To conclude, we can remark that since $q_1' = q'2$ the fact that $C[q_1'] \to^*_{\mathcal{S}_E(A)} q_1$ and $C[q_2'] \to^*_{\mathcal{S}_E(A)} q_2$ yields $q_1 = q_2$ by determinism of $\mathcal{S}_E(A)$.

- For the inductive step, let us assume that the property is true for any equivalence using k equations. For any equivalence using $k + 1$ equations, $s =_E s_1 =_E \ldots =_E s_k$, we can split this chain into two parts: $s =_E s_1$ using one equation and $s_1 =_E \ldots =_E s_k$ using k equations. From the first part, using case $k = 1$, we can deduce that s and s_1 are recognized by the same state q_1. From the second part, using induction hypothesis, we obtain that all terms $s_1, \ldots, s_k$ are recognized by the same state q_2. Finally from $s_1 \to^*_{\mathcal{S}_E(A)} q_1$ and $s_1 \to^*_{\mathcal{S}_E(A)} q_2$ and

determinism of $\mathcal{S}_E(A)$ we get that $q_1 = q_2$.

□

Now, the relative completeness result can be stated.

Theorem 2 (Criterion relative completeness)**.** *Let $\mathcal{F}$ an alphabet and E a set of equations on $\mathcal{F}$. If $\mathcal{T}(\mathcal{F})/_{=E}$ has a finite set of equivalence classes then EC-sequence $(A_i)^{i\geq 0}$ reaches a fixpoint A_k for $k \in \mathbb{N}$.*

Proof. We make a proof by contradiction. Assume that $\mathcal{T}(\mathcal{F})/_{=E}$ has a finite set of equivalence classes and that our criterion diverges, i.e., we build an infinite sequence $(A_i)^{i\geq 0}$. Let us denote by A_∞ the limit of this sequence. Note that, A_∞ has infinitely many states and transitions. By definition of the EC-sequence, A_∞ is in normal form w.r.t. $\mathcal{S}_E$. It is also complete since it recognizes all terms of $T_0, T_1, \ldots$ Since this automaton has an infinite number of states and the set of equivalence classes is finite, by the pigeonhole principle, we know that there exists two states q_1 and q_2 ($q_1 \neq q_2$) recognizing two terms in the same equivalence class. Let us call t_1 and t_2 those two terms s.t. $t_1 \rightarrow^*_{A_\infty} q_1$, $t_2 \rightarrow^*_{A_\infty} q_2$ and $t_1 =_E t_2$. Since A_∞ is in normal form w.r.t. $\mathcal{S}_E$ and is complete, by Lemma 2, we get that $q_1 = q_2$ which is a contradiction. □

Moreover, the states of the fixpoint automaton correspond exactly to the equivalence classes induced by E and $\mathcal{F}$.

Theorem 3 (One state per equivalence class)**.** *Let $\mathcal{F}$ an alphabet and E a set of equations on $\mathcal{F}$. If the EC-sequence $(A_i)^{i\geq 0}$ reaches a fixpoint A_k for $k \in \mathbb{N}$, then each equivalence class of $\mathcal{T}(\mathcal{F})/_{=E}$ is recognized by a unique and distinct state of A_k.*

Proof. Since A_k is complete, we know that there is *at least* one state to recognize any term of an equivalence class. On the other hand, since A_k is E-saturated, we know that all E-equivalent terms are recognized by the same state, so there is *at most* one state to recognize an equivalence class. □

8. Example and discussion

Let us illustrate our approach with an example. Contrary to Example 6, where the *EC*-sequence does not converge, we now present an example in which the process terminates and the final automaton exactly recognizes the equivalences classes. Let $E = \{f(x, f(y, z)) = f(f(x, y), z),\ f(x, f(y, z)) = f(y, z),\ f(x, y) = f(x, z)\}$, where $x, y, z \in \mathcal{X}$, and let $\mathcal{F} = \{a\!:\!0,\ b\!:\!0,\ f\!:\!2\}$. To prove that $\mathcal{T}(\mathcal{F})/_{=E}$ is finite, we cannot exploit techniques presented at the begining of Section 3. First, $\mathcal{R}$ is not ground. Second, the set E cannot be oriented into a TRS. In particular, the third equation cannot be oriented to the right because, for the rewrite rule to be valid, variables of the right-hand side of the equation need to be contained in the set of variables of the left-hand side. It cannot be oriented to the left for a symetrical reason. As a consequence, there exists no term rewriting system to decide equality on this particular set E. Finally, this set of equations does not belong to the class of shallow theories [4], since some variables in equations from E occur at depth two.

Example 7. *Here is a complete run of our criterion on the set E defined above. The set of terms of depth one is $T^1 = \{a, b\}$. According to Definition 13, the resulting automaton is $A_1 = \langle \mathcal{F}, Q, \Delta_1 \rangle$, with $Q = \{q_b, q_a\}$ and $\Delta_1 = \{a \to q_a,\ b \to q_b\}$. Note that the simplification step does not modify this automaton, as none of the equations in E can be applied at this stage.*

The algorithm then proceeds with $T^2 = \{f(t, t') \mid t, t' \in T^1\}$. According to Definitions 13 and 12, after the normalization step, we obtain the automaton $A_1 \cup \{f(q_a, q_b) \to q_{f(a,b)},\ f(q_a, q_a) \to q_{f(a,a)},\ f(q_b, q_b) \to q_{f(b,b)},\ f(q_b, q_a) \to q_{f(b,a))}\}$.

Then, the simplification step merges $q_{f(a,a)}$ and $q_{f(a,b)}$ into a single state $q_{f(a,a)}$. Similarly, $q_{f(b,a)}$ and $q_{f(b,b)}$ are merged into a single state $q_{f(b,a)}$. Both mergings are a consequence of the equation $f(x, y) = f(x, z)$, that implies that two f-terms are equal whenever their first arguments are equal.

$A_2 = A_1 \cup \{f(q_a, q_b) \to q_{f(a,a)},\ f(q_a, q_a) \to q_{f(a,a)},\ f(q_b, q_b) \to q_{f(b,a)},\ f(q_b, q_a) \to q_{f(b,a)}\}$.

Then, with $T^3 = \{f(t, t') \mid t, t' \in T^2\}$ and normalization we get:

$A_2 \cup \{f(q_{f(a,a)}, q_{f(a,a)}) \to q_{f(f(a,a),f(a,a))},\ f(q_{f(a,a)}, q_{f(b,a)}) \to q_{f(f(a,a),f(b,a))},$
$f(q_{f(b,a)}, q_{f(a,a)}) \to q_{f(f(b,a),f(a,a))},\ f(q_{f(b,a)}, q_{f(b,a)}) \to q_{f(f(b,a),f(b,a))}\}$.

For the same reason as in the previous step, $q_{f(f(a,a),f(b,a))}$ *and* $q_{f(f(a,a),f(a,a))}$ *are merged into* $q_{f(f(a,a),f(a,a))}$*, and* $q_{f(f(b,a),f(a,a))}$ *and* $q_{f(f(b,a),f(b,a))}$ *are merged into* $q_{f(f(b,a),f(a,a))}$*. This results into:*

$A_2 \cup \{f(q_{f(a,a)}, q_{f(a,a)}) \to q_{f(f(a,a),f(a,a))},\ f(q_{f(a,a)}, q_{f(b,a)}) \to q_{f(f(a,a),f(a,a))},$
$f(q_{f(b,a)}, q_{f(a,a)}) \to q_{f(f(b,a),f(a,a))},\ f(q_{f(b,a)}, q_{f(b,a)}) \to q_{f(f(b,a),f(a,a))}\}$.

Unlike in the previous step, the height of the terms in T^3 *now allows the application of the equation* $f(x, f(y, z)) = f(y, z)$*. Finally, the states* $q_{f(b,a)}$*,* $q_{f(a,a)}$*,* $q_{f(f(a,a),f(a,a))}$ *and* $q_{f(f(b,a),f(a,a))}$ *are all merged into the single state* $q_{f(f(b,a),f(a,a))}$*. The simplification step thus results in the automaton* A_3 *whose transitions are:*

$\{f(q_b, q_b) \to q_{f(f(b,a),f(a,a))},\ f(q_b, q_a) \to q_{f(f(b,a),f(a,a))},\ b \to q_b,\ f(q_a, q_b) \to q_{f(f(b,a),f(a,a))},\ f(q_{f(f(b,a),f(a,a))}, q_{f(f(b,a),f(a,a))}) \to q_{f(f(b,a),f(a,a))},\ f(q_a, q_a) \to q_{f(f(b,a),f(a,a))},\ a \to q_a\}$.

Observe that $T^4 = \{f(t, t') \mid t, t' \in T^3\}$ *is already covered by* A_3*. This indicates that a fixpoint has been reached. There are three equivalence classes, namely:* $\{a\}$*,* $\{b\}$*, and* $\{f(t_1, t_2) \mid t_1, t_2 \in \mathcal{T}(\mathcal{F})\}$.

In conclusion, considering E*, for any term* t*, we have* $t =_E a$*, or* $t =_E b$*, or* $t =_E f(b, b)$*, where* $f(b, b)$ *is a representative of the state* $q_{f(f(b,a),f(a,a))}$.

Determining whether $\mathcal{T}(\mathcal{F})/_{=E}$ is finite is a cornerstone of verification techniques based on rewriting techniques, tree automata and sets of equations [5, 6]. Finiteness of $\mathcal{T}(\mathcal{F})/_{=E}$ entails termination of the verification process. Using Theorems 1 and 2, if $\mathcal{T}(\mathcal{F})/_{=E}$ is finite then our criterion proves it, and guarantees termination of verification procedures of [5, 6].

Acknowledgements

Many thanks to Sophie Tison for the fruitful discussions and for pointing out the undecidability result on group theory used in Section 3.

References

[1] Bachmair, L., Tiwari, A., Vigneron, L., 2003. Abstract Congruence Closure. JAR 31, 129–168.

[2] Comon, H., 1986. Sufficient Completness, Term Rewriting Systems and "Anti-Unification", in: CADE'86, Springer. pp. 128–140.

[3] Comon, H., Dauchet, M., Gilleron, R., Lugiez, D., Tison, S., Tommasi, M., 2007. *Tree Automata Techniques and Applications* (TATA).

[4] Comon, H., Haberstrau, M., Jouannaud, J.P., 1994. Syntacticness, Cycle-syntacticness, and Shallow theories. Inf. Comput. 111, 154–191.

[5] Genet, T., 2016. Termination Criteria for Tree Automata Completion. Journal of Logical and Algebraic Methods in Programming 85, Issue 1, Part 1, 3–33.

[6] Genet, T., Haudebourg, T., Jensen, T., 2018. Verifying higher-order functions with tree automata, in: FoSSaCS'18, Springer.

[7] Genet, T., Rusu, R., 2010. Equational tree automata completion. Journal of Symbolic Computation 45, 574–597.

[8] Knuth, D.E., Bendix, P.B., 1983. Simple Word Problems in Universal Algebras. Springer Berlin Heidelberg. pp. 342–376.

[9] Lyndon, R.C., Schupp, P.E., 1977. Combinatorial Group Theory. volume 89. Springer.

[10] Otto, F., 1998. Some undecidability results concerning the property of preserving regularity. TCS 207, 43–72.