Reachability Analysis of Innermost Rewriting

Thomas Genet, Yann Salmon

Reachability Analysis of Innermost Rewriting

Thomas Genet* Yann Salmon†

February 10, 2014

Approximating the set of terms reachable by rewriting finds more and more applications ranging from termination proofs of term rewriting systems, cryptographic protocol verification to static analysis of programs. However, since approximation techniques do not take rewriting strategies into account, they build very coarse approximations when rewriting is constrained by a specific strategy. In this work, we propose to adapt the Tree Automata Completion algorithm to accurately approximate the set of terms reachable by rewriting under the innermost strategy. We prove that the proposed technique is sound and precise w.r.t. innermost rewriting. The proposed algorithm has been implemented in the Timbuk reachability tool. Experiments shows that it noticeably improves the accuracy of static analysis for functional programs using the call-by-value evaluation strategy. In particular, for some functional programs needing lazy evaluation to terminate, the computed approximations are precise enough to prove the absence of innermost normal forms, *i.e.* prove non termination of the program with call-by-value.

1. Introduction

For a Term Rewriting System (TRS) R and a set of terms $L_0 \subseteq T(\Sigma)$, the set of reachable terms is $R^*(L_0) = \left\{t \in T(\Sigma) \,\middle|\, \exists s \in L_0, s \to_R^* t\right\}$. This set can be computed for specific classes of R but, in general, it has to be approximated. Applications of the approximation of $R^*(L_0)$ are ranging from cryptographic protocol verification [GK00, ABB+05], to static analysis of various programming languages [BGJL07, KO11] or to TRS termination proofs [Mid02, GHWZ05]. Most of the techniques compute such approximations using tree automata as the core formalism to represent or approximate the (possibly) infinite set of terms $R^*(L_0)$. Most of them also rely on a Knuth-Bendix completion-like algorithm to produce an automaton $\mathcal{A}^*$ recognising exactly, or over-approximating, the set of reachable terms. As a result, these techniques can be refered as *tree automata completion* techniques [Gen98, TKS00, Tak04, FGVTT04, BCHK09, GR10, Lis12].

Surprisingly, very little effort has been paid to computing or over-approximating $R^*_{strat}(L_0)$, i.e. set of reachable terms when R is applied with a strategy $strat$. To the best of our knowledge, Pierre Réty and Julie Vuotto's work [RV02] is the first one to have tackled this goal. They give some sufficient conditions on L_0 and R for $R^*_{strat}(L_0)$ to be recognised by a tree automaton $\mathcal{A}^*$,

*I.R.I.S.A., Campus de Beaulieu, 35042 Rennes Cedex, France, Thomas.Genet@irisa.fr
†I.R.I.S.A., Campus de Beaulieu, 35042 Rennes Cedex, France, Yann.Salmon@irisa.fr

where *strat* can be the innermost or the outermost strategy. Innermost reachability for shallow TRSs was studied in [GGJ08]. However, in both cases, those restrictions on R are strong and generally incompatible with programs seen as TRS. Moreover, the proposed techniques are not able to over-approximate reachable terms when the TRS are outside of this class.

In this paper, we define a tree automata completion algorithm over-approximating the set $R^*_{\text{in}}(L_0)$ for all left-linear TRSs R and all regular set of input terms L_0. As the completion algorithm of [GR10], it is parameterised by a set of term equations E defining the precision of the approximation. We prove the soundness of the algorithm: for all set of equation E, if completion terminates then the resulting automaton $\mathcal{A}^*$ recognises an over-approximation of $R^*_{\text{in}}(L_0)$. Then, we prove a precision theorem: $\mathcal{A}^*$ recognises no more terms that the set of terms reachable by innermost rewriting with R modulo equations of E. Finally, we show on a simple example that the precision of this innermost completion can improve a lot the accuracy of the analysis for functional programs.

This paper is organised as follows. Section 2 recalls some basic notions about TRSs and tree automata. Section 3 defines tree automata completion. Section 4 explains how to adapt completion so as to take innermost strategy into account. Section 5 states and proves the soundness of this method. Section 6 demonstrates how our new technique can effectively give more precise results on functional programs thanks to the tool TimbukSTRAT, an implementation of our method in the Timbuk reachability tool [Tim14]. Section 7 states a precision theorem.

2. Basic notions and notations

2.1. Terms

Definition 1 (Signature).
A signature is a set whose elements are called function symbols. Each function symbol has an arity, which is a natural integer. Function symbols of arity 0 are called constants. Given a signature Σ and $k \in \mathbb{N}$, the set of its function symbols of arity k is noted Σ_k. 1 ◀

Definition 2 (Term, ground term, linearity).
Given a signature Σ and a set $\mathcal{X}$ whose elements are called variables and such that $\Sigma \cap \mathcal{X} = \varnothing$, we define the set of terms over Σ and $\mathcal{X}$, $T(\Sigma, \mathcal{X})$, as the smallest set such that :

1. $\mathcal{X} \subseteq T(\Sigma, \mathcal{X})$ and
2. $\forall k \in \mathbb{N}, \forall f \in \Sigma_k, \forall t_1, \ldots, t_k \in T(\Sigma, \mathcal{X}), f(t_1, \ldots, t_k) \in T(\Sigma, \mathcal{X})$.

Terms in which no variable appears, i.e. terms in $T(\Sigma, \varnothing)$, are called ground; the set of ground terms is noted $T(\Sigma)$.
Terms in which any variable appears at most once are called linear.[1] 2 ◀

Definition 3 (Substitution).
A substitution over $T(\Sigma, \mathcal{X})$ is an application from $\mathcal{X}$ to $T(\Sigma, \mathcal{X})$. Any substitution is inductively extended to $T(\Sigma, \mathcal{X})$ by $\sigma(f(t_1, \ldots, t_k)) = f(\sigma(t_1), \ldots, \sigma(t_k))$. Given a substitution σ and a term t, we note $t\sigma$ instead of $\sigma(t)$. 3 ◀

[1] In particular, any ground term is linear.

Definition 4 (Context).
A context over $T(\Sigma,\mathcal{X})$ is a term in $T(\Sigma \cup \mathcal{X}, \{\Box\})$ in which the variable $\Box$ appears exactly once. A ground context over $T(\Sigma,\mathcal{X})$ is a context over $T(\Sigma)$. The smallest possible context, $\Box$, is called the trivial context. Given a context C and a term t, we note $C[t]$ the term $C\sigma_t$, where $\sigma_t : \Box \mapsto t$. 4 ◀

Definition 5 (Position).
Positions are finite words over the alphabet $\mathbb{N}$. The set of positions of term t, $\mathrm{Pos}(t)$, is defined by induction over t:

1. for all constant c and all variable X, $\mathrm{Pos}(c) = \mathrm{Pos}(X) = \{\Lambda\}$ and
2. $\mathrm{Pos}(f(t_1,\dots,t_k)) = \{\Lambda\} \cup \bigcup_{i=1}^{k} \{i\}.\mathrm{Pos}(t_i)$. 5 ◀

Definition 6 (Subterm-at-position, replacement-at-position).
The position of the hole in context C, $\mathrm{Pos}_\Box(C)$, is defined by induction on C:

1. $\mathrm{Pos}_\Box(\Box) = \Lambda$
2. $\mathrm{Pos}_\Box(f(C_1,\dots,C_k)) = i.\mathrm{Pos}_\Box(C_i)$, where i is the unique integer in $[\![1\,;k]\!]$ such that C_i is a context.

Given a term u and $p \in \mathrm{Pos}(u)$, there is a unique context C and a unique term v such that $\mathrm{Pos}_\Box(C) = p$ and $u = C[v]$. The term v is noted $u_{|p}$, and, given another term t, we note $u[t]_p = C[t]$. 6 ◀

2.2. Rewriting

Definition 7 (Rewriting rule, term rewriting system).
A rewriting rule over $(\Sigma,\mathcal{X})$ is a couple $(\ell, r) \in T(\Sigma,\mathcal{X}) \times T(\Sigma,\mathcal{X})$, that we note $\ell \to r$, such that any variable appearing in r also appears in ℓ. A term rewriting system (TRS) over $(\Sigma,\mathcal{X})$ is a set of rewriting rules over $(\Sigma,\mathcal{X})$. 7 ◀

Definition 8 (Rewriting step, redex, reducible term, normal form).
Given a signature $(\Sigma,\mathcal{X})$, a TRS R over it and two terms $s,t \in T(\Sigma)$, we say that s can be rewritten into t by R, and we note $s \to_R t$ if there exist a rule $\ell \to r \in R$, a ground context C over $T(\Sigma)$ and a substitution σ over $T(\Sigma,\mathcal{X})$ such that $s = C[\ell\sigma]$ and $t = C[r\sigma]$.
In this situation, the term s is said to be reducible by R and the subterm $\ell\sigma$ is called a redex of s. A term s that is not reducible by R is a normal form of R. The set of normal forms of R is noted $\textsc{Irr}(R)$.
We note $\to_R^*$ the reflexive and transitive closure of $\to_R$. 8 ◀

Definition 9 (Set of reachable terms).
Given a signature $(\Sigma,\mathcal{X})$, a TRS R over it and a set of terms $L \subseteq T(\Sigma)$, we note $R(L) = \{t \in T(\Sigma) \mid \exists s \in L, s \to_R t\}$ and $R^*(L) = \left\{t \in T(\Sigma) \,\middle|\, \exists s \in L, s \to_R^* t\right\}$. 9 ◀

Definition 10 (Left-linearity).
A TRS R is said to be left-linear if for each rule $\ell \to r$ of R, the term ℓ is linear. 10 ◀

Definition 11 (Constructors and defined symbols, sufficient completeness).
Given a TRS R over $(\Sigma, \mathcal{X})$, there is a partition $(\mathcal{C}, \mathcal{D})$ of Σ such that all symbols occurring at the root position of left-hand sides of rules of R are in $\mathcal{D}$. $\mathcal{D}$ is the set of defined symbols of R, $\mathcal{C}$ is the set of constructors. Terms in $T(\mathcal{C})$ are called data-terms. A TRS R over $(\Sigma, \mathcal{X})$ is sufficiently complete if for all $s \in T(\Sigma)$, $R^*(\{s\}) \cap T(\mathcal{C}) \neq \varnothing$. 11 ◀

2.3. Equations

Definition 12 (Equivalence relation, congruence).
A binary relation over some set S is an equivalence relation if it is reflexive, symmetric and transitive.

An equivalence relation $\equiv$ over $T(\Sigma)$ is a congruence if for all $k \in \mathbb{N}$, for all $f \in \Sigma_k$, for all $t_1, \ldots, t_k, s_1, \ldots, s_k \in T(\Sigma)$ such that $\forall i \in [\![1\,;k]\!]$, $t_i \equiv s_i$, we have $f(t_1, \ldots, t_k) \equiv f(s_1, \ldots, s_k)$. 12 ◀

Definition 13 (Equation, $\equiv_E$).
An equation over $(\Sigma, \mathcal{X})$ is a pair of terms $(s, t) \in T(\Sigma, \mathcal{X}) \times T(\Sigma, \mathcal{X})$, that we note $s = t$. A set E of equations over $(\Sigma, \mathcal{X})$ induces a congruence $\equiv_E$ over $T(\Sigma)$ which is the smallest congruence over $T(\Sigma)$ such that for all $s = t \in E$ and for all substitution $\theta : \mathcal{X} \to T(\Sigma)$, $s\theta \equiv_E t\theta$. The classes of equivalence of $\equiv_E$ are noted with $[\cdot]_E$. 13 ◀

Definition 14 (Rewriting modulo E).
Given a TRS R and a set of equations E both over $(\Sigma, \mathcal{X})$, we define the R modulo E rewriting relation, $\to_{R/E}$, as follows. For any $u, v \in T(\Sigma)$, $u \to_{R/E} v$ if and only if there exist $u', v' \in T(\Sigma)$ such that $u' \equiv_E u$, $v' \equiv_E v$ and $u' \to_R v'$.

We define $\to^*_{R/E}$, $(R/E)(L)$ and $(R/E)^*(L)$ for $L \subseteq T(\Sigma)$ as in Definitions 8 and 9. 14 ◀

2.4. Tree automata

Definition 15 (Tree automaton, delta-transition, epsilon-transition, new state).
An automaton over Σ is some $\mathcal{A} = (\Sigma, Q, Q_F, \Delta)$ where Q is a finite set of states, Q_F is a subset of Q whose elements are called final states and Δ a finite set of transitions. A delta-transition is of the form $f(q_1, \ldots, q_k) \rightarrowtail q'$ where $f \in \Sigma_k$ and $q_1, \ldots, q_k, q' \in Q$. An epsilon-transition is of the form $q \rightarrowtail q'$ where $q, q' \in Q$. A configuration of $\mathcal{A}$ is a term in $T(\Sigma, Q)$.

A state $q \in Q$ that appears nowhere in Δ is called a new state. A configuration is elementary if each of its subconfigurations at depth 1 (if any) is a state. A configuration is trivial if it is just a state. 15 ◀

Remark. We simply write $\mathcal{A}$ to denote an automaton, write $Q_\mathcal{A}$ for the set of states of $\mathcal{A}$. We assimilate an automaton with its set of transitions. When taking a "new state", we silently expand $Q_\mathcal{A}$ if needed. We are only rarely interested in Q_F, the set of final states.

Definition 16.
Let $\mathcal{A} = (\Sigma, Q, Q_F, \Delta)$ be an automaton and let c, c' be configurations of $\mathcal{A}$. We say that $\mathcal{A}$ recognises c into c' in one step, and note $c \underset{\mathcal{A}}{\rightarrowtail} c'$ if there a transition $\tau \rightarrowtail \rho$ in $\mathcal{A}$ and a context C over $T(\Sigma, Q)$ such that $c = C[\tau]$ and $c' = C[\rho]$. We note $\overset{*}{\underset{\mathcal{A}}{\rightarrowtail}}$ the reflexive and transitive closure of $\underset{\mathcal{A}}{\rightarrowtail}$ and, for any $q \in Q$, $\mathscr{L}(\mathcal{A}, q) = \left\{ t \in T(\Sigma) \;\middle|\; t \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} q \right\}$. We extend this definition to subsets of Q and note $\mathscr{L}(\mathcal{A}) = \mathscr{L}(\mathcal{A}, Q_F)$. 16 ◀

Example 17.
Let Σ be defined with $\Sigma_0 = \{n, 0\}$, $\Sigma_1 = \{s, a, f\}$, $\Sigma_2 = \{c\}$ where 0 is meant to represent integer zero, s the successor operation on integers, a the predecessor ("antecessor") operation, n the empty list, c the constructor of lists of integers and f intended to be the function on lists that filters out integer zero. Let $R = \{f(n) \to n, f(c(s(X), Y)) \to c(s(X), f(Y)), f(c(a(X), Y)) \to c(a(X), f(Y)), f(c(0, Y)) \to f(Y), a(s(X)) \to X, s(a(X)) \to X\}$. Let $\mathcal{A}_0 = \{n \rightarrowtail q_n, 0 \rightarrowtail q_0, s(q_0) \rightarrowtail q_s, a(q_s) \rightarrowtail q_a, c(q_a, q_n) \rightarrowtail q_c, f(q_c) \rightarrowtail q_f\}$. We have $\mathscr{L}(\mathcal{A}_0, q_f) = \{f(c(a(s(0)), n))\}$ and $R(\mathscr{L}(\mathcal{A}_0, q_f)) = \{f(c(0, n)), c(a(s(0)), f(n))\}$. 17 ◁

Definition 18 (Determinism, Completeness, Accessibility).
An automaton is deterministic if it has no epsilon-transition and for all delta-transitions $\tau \rightarrowtail \rho$ and $\tau' \rightarrowtail \rho'$, if $\tau = \tau'$ then $\rho = \rho'$. An automaton is complete if each of its non-trivial configurations is the left-hand side of some of its transitions. A state q of automaton $\mathcal{A}$ is accessible if $\mathscr{L}(\mathcal{A}, q) \neq \varnothing$. An automaton is accessible if all of its states are. 18 ◀

Remark. Transitions may have "colours", like $\mathfrak{K}$ for transition $q \overset{\mathfrak{K}}{\rightarrowtail} q'$. We will use colours $\mathfrak{K}$ and $\mathfrak{E}$.

Definition 19.
Given an automaton $\mathcal{A}$ and a colour $\mathfrak{K}$, we denote by $\mathcal{A}^{\not\mathfrak{K}}$ the automaton obtained from $\mathcal{A}$ by removing all transitions coloured with $\mathfrak{K}$. 19 ◀

Definition 20.
Given two states q, q' of some automaton $\mathcal{A}$ and a colour $\mathfrak{E}$, we note $q \underset{\mathcal{A}}{\overset{\mathfrak{E}}{\rightleftarrows}} q'$ when we have both $q \underset{\mathcal{A}}{\overset{\mathfrak{E}}{\rightarrowtail}} q'$ and $q' \underset{\mathcal{A}}{\overset{\mathfrak{E}}{\rightarrowtail}} q$. 20 ◀

Remark. $q \underset{\mathcal{A}}{\overset{\mathfrak{E},*}{\rightleftarrows}} q'$ is stronger than $(q \underset{\mathcal{A}}{\overset{\mathfrak{E},*}{\rightarrowtail}} q' \wedge q' \underset{\mathcal{A}}{\overset{\mathfrak{E},*}{\rightarrowtail}} q)$. $\underset{\mathcal{A}}{\overset{\mathfrak{E},*}{\rightleftarrows}}$ is an equivalence relation over $Q_{\mathcal{A}}$. This relation is extended to a congruence relation over $T(\Sigma, Q)$. The equivalence classes are noted with $[\cdot]_{\mathfrak{E}}$.

Definition 21.
Let $\mathcal{A} = (\Sigma, Q, Q_F, \Delta)$ be an automaton and $\mathfrak{E}$ a colour. We note $\mathcal{A}/\mathfrak{E}$ the automaton over Σ whose set of states is $Q/\mathfrak{E}$, whose set of final states if $Q_F/\mathfrak{E}$ and whose set of transitions is

$$\left\{f([q_1]_{\mathfrak{E}}, \ldots, [q_k]_{\mathfrak{E}}) \rightarrowtail [q']_{\mathfrak{E}} \;\middle|\; f(q_1, \ldots, q_k) \rightarrowtail q' \in \Delta\right\}$$
$$\cup \left\{[q]_{\mathfrak{E}} \rightarrowtail [q']_{\mathfrak{E}} \;\middle|\; q \rightarrowtail q' \in \Delta \wedge [q]_{\mathfrak{E}} \neq [q']_{\mathfrak{E}}\right\}.$$
21 ◀

Remark. For any configurations c, c' of $\mathcal{A}$, we have $c \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} c'$ if and only if $[c]_{\mathfrak{E}} \underset{\mathcal{A}/\mathfrak{E}}{\overset{*}{\rightarrowtail}} [c']_{\mathfrak{E}}$. So the languages recognised by $\mathcal{A}$ and $\mathcal{A}/\mathfrak{E}$ are the same.

We now give notations used for pair automata, the archetype of which is the product of two automata.

Definition 22 (Pair automaton).
An automaton $\mathcal{A} = (\Sigma, Q, Q_F, \Delta)$ is said to be a pair automata if there exists some sets Q_1 and Q_2 such that $Q = Q_1 \times Q_2$. 22 ◀

Definition 23 (Product automaton).
Let $\mathcal{A} = (\Sigma, Q, Q_F, \Delta_\mathcal{A})$ and $\mathcal{B} = (\Sigma, P, P_F, \Delta_\mathcal{B})$ be two automata. The product automaton of $\mathcal{A}$ and $\mathcal{B}$ is $\mathcal{A} \times \mathcal{B} = (\Sigma, Q \times P, Q_F \times P_F, \Delta)$ where

$$\Delta = \left\{ f(\langle q_1, p_1 \rangle, \ldots, \langle q_k, p_k \rangle) \rightarrowtail \langle q', p' \rangle \;\middle|\; f(q_1, \ldots, q_k) \rightarrowtail q' \in \Delta_\mathcal{A} \wedge f(p_1, \ldots, p_k) \rightarrowtail p' \in \Delta_\mathcal{B} \right\}$$
$$\cup \left\{ \langle q, p \rangle \rightarrowtail \langle q', p' \rangle \;\middle|\; q \rightarrowtail q' \in \Delta_\mathcal{A} \right\} \cup \left\{ \langle q, p \rangle \rightarrowtail \langle q, p' \rangle \;\middle|\; p \rightarrowtail p' \in \Delta_\mathcal{B} \right\}.$$ 23 ◀

Definition 24 (Projections).
Let $\mathcal{A} = (\Sigma, Q, Q_F, \Delta)$ be a pair automaton, let $\tau \rightarrowtail \rho$ be one of its transitions and $\langle q, p \rangle$ be one of its states. We define $\Pi_1(\langle q, p \rangle) = q$ and extend $\Pi_1(\cdot)$ to configurations inductively: $\Pi_1(f(\gamma_1, \ldots, \gamma_k)) = f(\Pi_1(\gamma_1), \ldots, \Pi_1(\gamma_k))$. We define $\Pi_1(\tau \rightarrowtail \rho) = \Pi_1(\tau) \rightarrowtail \Pi_1(\rho)$. We define $\Pi_1(\mathcal{A}) = (\Sigma, \Pi_1(Q), \Pi_1(Q_F), \Pi_1(\Delta))$. $\Pi_2(\cdot)$ is defined on all these objects in the same way for the right components. 24 ◀

Remark. Using $\Pi_1(\mathcal{A})$ amounts to forgetting the precision given by the right components of the states. As a result, $\mathscr{L}(\Pi_1(\mathcal{A}), q) \supseteq \mathscr{L}(\mathcal{A}, \langle q, p \rangle)$.

2.5. Innermost strategies

In general, a strategy over a TRS R is a set of (computable) criteria to describe a certain subrelation of $\rightarrow_R$. In this paper, we will be interested in innermost strategies. In these strategies, commonly used to execute functional programs ("call-by-value"), terms are rewritten by always contracting one of the lowest reducible subterms.

Definition 25 (Innermost strategy).
Given a TRS R and two terms s, t, we say that s can be rewritten into t by R with an innermost strategy, and we note $s \rightarrow_{R_{\text{in}}} t$, if $s \rightarrow_R t$ and each strict subterm of the redex in s is a R-normal form. We define $\rightarrow^*_{R_{\text{in}}}$, $R_{\text{in}}(L)$ and $R^*_{\text{in}}(L)$ as in Definitions 8 and 9. 25 ◀

Example 26.
Let us go on with Example 17: $R_{\text{in}}(\mathscr{L}(\mathcal{A}_0, q_f)) = \{f(c(0, n))\}$. 26 ◁

Remark. It is in fact sufficient to check whether subterms of s at depth 1 are in normal form to decide whether s can be rewritten with an innermost strategy.

To deal with innermost strategies, we will have to discriminate normal forms. This is possible within the tree automaton framework when R is left-linear.

Theorem 27 ([CR87]).
Let R be a left-linear TRS. There is a deterministic and complete tree automaton $\mathcal{IRR}(R)$ whose states are all final except one, noted p_{red} and such that $\mathscr{L}(\mathcal{IRR}(R)) = \text{IRR}(R)$ and $\mathscr{L}(\mathcal{IRR}(R), p_{\text{red}}) = T(\Sigma) \setminus \text{IRR}(R)$. 27 ■

Remark. From determinism and the property of p_{red} follows that for any state p different from p_{red}, $\mathscr{L}(\mathcal{IRR}(R), p) \subseteq \text{IRR}(R)$.

Remark. If a term is reducible, any term built upon it is also reducible. Thus any transition of $\mathcal{IRR}(R)$ where p_{red} appears in the left-hand side will necessarily have p_{red} as its right-hand side. Therefore, for brevity, these transitions will always be left implicit when describing the automaton $\mathcal{IRR}(R)$ for some TRS R.

Example 28.

In Example 17, $\mathcal{IRR}(R)$ needs, in addition to p_{red}, a state p_{list} to recognise lists of integers, a state p_a for terms of the form $a(\dots)$, a state p_s for $s(\dots)$, a state p_0 for 0 and a state p_{var} to recognise terms that are not subterms of lhs of R, but may participate in building a reducible term by being instances of variables in a lhs. We note $P = \{p_{list}, p_0, p_a, p_s, p_{var}\}$ and $P_{int} = \{p_0, p_a, p_s\}$. The interesting transitions are thus $0 \rightarrowtail p_0$, $\bigcup_{p \in P \setminus \{p_a\}} \{s(p) \rightarrowtail p_s\}$, $\bigcup_{p \in P \setminus \{p_s\}} \{a(p) \rightarrowtail p_a\}$; $n \rightarrowtail p_{list}$, $\bigcup_{p \in P_{int}, p' \in P} \{c(p,p') \rightarrowtail p_{list}\}$; $f(p_{list}) \rightarrowtail p_{\text{red}}$, $a(p_s) \rightarrowtail p_{\text{red}}$, $s(p_a) \rightarrowtail p_{\text{red}}$. Furthermore, as remarked above, any configuration that contains a p_{red} is recognised into p_{red}. Finally, some configurations are not covered by the previous cases: they are recognised into p_{var}. 28 ◁

3. Classical equational completion

Equational completion of [GR10] is an iterative process on automata that is not guaranteed to terminate. Each iteration comprises two parts: (exact) completion itself, then equational merging. The former tends to incorporate descendants by R of already recognised terms into the recognised language; this leads to the creation of new states. The latter tends to merge states in order to ease termination of the overall process, at the cost of precision of the computed result. Some transition added by equational completion will have colours $\mathfrak{R}$ or $\mathfrak{E}$; it is assumed that the transitions of the input automaton $\mathcal{A}_0$ do not have any colour and that $\mathcal{A}_0$ does not have any epsilon-transition.

3.1. Exact completion

Exact completion is about resolving critical pairs. A critical pair represents a situation where some term is recognised by the current automaton, but not its descendants by R. Its resolution consists in adding transitions to let the descendants be recognised as well. This process can create new critical pairs.

Definition 29 (Critical pair).

A pair $(\ell \to r, \sigma, q)$ where $\ell \to r \in R$, $\sigma : \mathcal{X} \to Q_{\mathcal{A}}$ and $q \in Q_{\mathcal{A}}$ is critical if (see Figure 1(a))

1. $\ell\sigma \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} q$ and
2. $r\sigma \overset{*}{\underset{\mathcal{A}}{\not\rightarrowtail}} q$. 29 ◀

(a) A critical pair

(b) Situation of application

Figure 1: A critical pair and a situation of application of an equation

We will want to close the square of Figure 2, that is to add transitions $r\sigma \rightarrowtail q'$ and $q' \rightarrowtail q'$. However, doing the former is not generally possible in one step, as $r\sigma$ might be a non-elementary configuration. We will have to take steps and maybe introduce new states: this is what normalisation does. Given an automaton $\mathcal{A}$, a configuration c of $\mathcal{A}$ and a new state q, we note $Norm_{\mathcal{A}}(c,q)$ the set of transitions (with new states) that we add to $\mathcal{A}$ to ensure that c is recognised into q. The formal definition of normalisation is presented in Appendix A.

Example 30.

With a suitable signature, suppose that automaton $\mathcal{A}$ consists of the transitions $c \rightarrowtail q_1$ and $f(q_1) \rightarrowtail q_2$ and we want to normalise $f(g(q_2,c))$ to the new state q_N. We first have to normalise under g: q_2 is already a state, so it does not need to be normalised; c has to be normalised to a state: since $\mathcal{A}$ already has transition $c \rightarrowtail q_1$, we add nothing and it remains to normalise $g(q_2,q_1)$. Since $\mathcal{A}$ does not contain a transition for this configuration, we must add a new state q' and the transition $g(q_2,q_1) \rightarrowtail q'$. Finally, we add $f(q') \rightarrowtail q_N$. 30 ◁

Definition 31 (Completion of a critical pair).

A critical pair $CP = (\ell \rightarrow r, \sigma, q)$ in automaton $\mathcal{A}$ is completed by first computing $N = Norm_{\mathcal{A}^{\mathcal{R}}}(r\sigma, q')$ where q' is a new state, then adding to $\mathcal{A}$ the new states and the transitions appearing in N as well as the transition $q' \overset{\mathcal{R}}{\rightarrowtail} q$. If $r\sigma$ is a trivial configuration (ie. r is just a variable), only transition $r\sigma \overset{\mathcal{R}}{\rightarrowtail} q$ is added. 31 ◀

Definition 32 (Step of completion).

Let PC be the set of all critical pairs of $\mathcal{A}_i$. For $pc \in PC$, let N_{pc} be the set of new states and transitions needed under Definition 31 to complete pc. Then $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \bigcup_{pc \in PC} N_{pc}$. 32 ◀

Example 33.

Let Σ be defined with $\Sigma_0 = \{n, 0\}$, $\Sigma_1 = \{s, a, f\}$, $\Sigma_2 = \{c\}$ where 0 is meant to represent integer zero, s the successor operation on integers, a the predecessor ("antecessor") operation, n the empty list, c the constructor of lists of integers and f intended by some unwise person to be the function on lists that filters out integer zero. Let $R = \{f(n) \rightarrow n, f(c(s(X),Y)) \rightarrow c(s(X), f(Y)), f(c(a(X),Y)) \rightarrow c(a(X), f(Y)), f(c(0,Y)) \rightarrow f(Y), a(s(X)) \rightarrow X, s(a(X)) \rightarrow X\}$. Let $\mathcal{A}_0 = \{n \rightarrowtail q_n, 0 \rightarrowtail q_0, s(q_0) \rightarrowtail q_s, a(q_s) \rightarrowtail q_a, c(q_a,q_n) \rightarrowtail q_c, f(q_c) \rightarrowtail q_f\}$.
We have $\mathscr{L}(\mathcal{A}_0, q_f) = \{f(c(a(s(0)),n))\}$ and $R(\mathscr{L}(\mathcal{A}_0, q_f)) = \{f(c(0,n)), c(a(s(0)), f(n))\}$.

There is a critical pair CP_1 in $\mathcal{A}_0$ with the rule $f(c(a(X),Y)) \rightarrow c(a(X), f(Y))$, the substitution $\sigma_1 = \{X \mapsto q_s, Y \mapsto q_n\}$ and the state q_f. It is resolved by adding transitions to recognise $c(a(q_s), f(q_n))$ into q_f. Normalisation finds and reuses the transition $a(q_s) \rightarrowtail q_a$. It has to create a new state q_{N1} such that $f(q_n) \rightarrowtail q_{N1}$, and q_{N2} such that $c(q_a, q_{N1}) \rightarrowtail q_{N2}$. We then add $q_{N2} \overset{\mathcal{R}}{\rightarrowtail} q_f$, and have produced $\mathcal{J}_R^{(PC_1)}(\mathcal{A}_0)$.

Another critical pair is CP_2 in $\mathcal{A}_0$ with the rule $a(s(X)) \rightarrow X$, the substitution $\sigma_2 = \{X \mapsto q_0\}$ and the state q_a. It is resolved by adding to $\mathcal{J}_R^{(PC_1)}(\mathcal{A}_0)$ the transition $q_0 \overset{\mathcal{R}}{\rightarrowtail} q_a$, producing $\mathcal{J}_R^{(PC_1,PC_2)}(\mathcal{A}_0)$.

There is no more critical pair in $\mathcal{A}_0$: thus $\mathcal{C}_R(\mathcal{A}_0) = \mathcal{J}_R^{(PC_1,PC_2)}(\mathcal{A}_0)$. There is a new critical pair in $\mathcal{C}_R(\mathcal{A}_0)$ with $f(n) \rightarrow n$, the empty substitution and state q_{N1}. 33 ◁

3.2. Equational merging

Since completion of a critical pair can create new critical pairs, the process fuels itself, which is problematic for obtaining a fix-point. Equational merging is a way of countering this phenomenon at the cost of precision that is parametrised by equations over $T(\Sigma)$.

Definition 34 (Situation of application of an equation).
Given an equation $s = t$, an automaton $\mathcal{A}$, a $\theta : \mathcal{X} \to Q_{\mathcal{A}}$ and states q_1 and q_2, we say that $(s = t, \theta, q_1, q_2)$ is a situation of application in $\mathcal{A}$ if (see Figure 1(b))

1. $s\theta \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} q_1$,
2. $t\theta \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} q_2$ and
3. $q_1 \overset{\text{Ɛ}}{\underset{\mathcal{A}}{\not\rightarrowtail}} q_2$.

34 ◀

Definition 35 (Application of an equation).
Given $(s = t, \theta, q_1, q_2)$ a situation of application in $\mathcal{A}$, applying the underlying equation in it consists in adding transitions $q_1 \overset{\text{Ɛ}}{\rightarrowtail} q_2$ and $q_2 \overset{\text{Ɛ}}{\rightarrowtail} q_1$ to $\mathcal{A}$. This produces a new automaton $\mathcal{A}'$ and we note $\mathcal{A} \rightsquigarrow_E \mathcal{A}'$.

35 ◀

Remark. In [GR10], q_1 and q_2 were merged by "renaming" q_2 into q_1, ie. removing q_2 from $Q_{\mathcal{A}}$ and replacing every occurrence of q_2 by q_1 in the transitions of $\mathcal{A}$. This is equivalent to applying our method, then considering automaton $\mathcal{A}/\text{Ɛ}$ (see definition 21) and finally choosing a representative (here q_1 for the class $\{q_1, q_2\}$) of each equivalence class of states.

Definition 36 (Simplified automaton).
Given an automaton $\mathcal{A}$ and a set of equations E, we call simplified automaton of $\mathcal{A}$ by E and note $\mathcal{S}_E(\mathcal{A})$ the automaton resulting from the successive application of all applicable equations in $\mathcal{A}$.

36 ◀

Remark ($\rightsquigarrow_E$ is confluent). Indeed, there is a unique automaton $\mathcal{A}^!$ that differs from $\mathcal{A}$ only by its Ɛ-transitions and is such $\mathcal{A}^{\not\text{Ɛ}} \rightsquigarrow_E^* (\mathcal{A}^!)^{\not\text{Ɛ}}$ and there is no more situation of application of equations in $(\mathcal{A}^!)^{\not\text{Ɛ}}$.

Definition 37 (Step of equational completion).
A step of equational completion is the composition of a step of exact completion, then equational simplification: $\mathcal{CE}_{R,E}(\mathcal{A}) = \mathcal{S}_E(\mathcal{C}_R(\mathcal{A}))$.

37 ◀

The following notion is part of an easier discourse about the R/E-coherence notion of [GR10].

Definition 38 (Coherent automaton).
Let $\mathcal{A}_0 = (\Sigma, Q, Q_F, \Delta)$ be a tree automaton and E a set of equations. The automaton $\mathcal{A}_0$ is said to be *E-coherent* if for all $q \in Q$, there exists $s \in \mathscr{L}(\mathcal{A}_0, q)$ such that $\mathscr{L}(\mathcal{A}_0, q) \subseteq [s]_E$.

38 ◀

3.3. Known results

We now recall the two main theorems of [GR10].

Theorem 39 (Correctness).
Let $\mathcal{A}_0$ be some automaton. Assume the equational completion procedure defined above terminates when applied to $\mathcal{A}_0$. Let $\mathcal{A}_*$ be the resulting fix-point automaton. If R is left-linear, then the calculated over-approximation is correct, that is

$$\mathscr{L}(\mathcal{A}_*) \supseteq R^*(\mathscr{L}(\mathcal{A}_0)).$$ 39 ■

We will make usage of E-coherence for the precision theorem.

Lemma 40.
Let $\mathcal{A}_0$ be a E-coherent automaton, R a left-linear TRS and $\mathcal{A}$ be an automaton obtained from $\mathcal{A}_0$ after several steps of equational completion with R, E. Then $\mathcal{A}^{\not\Re}$ is E-coherent and moreover, for all state q of $\mathcal{A}$, there exists $s \in \mathscr{L}(\mathcal{A}^{\not\Re}, q)$ such that $\mathscr{L}(\mathcal{A}, q) \subseteq (R/E)^*(s)$. 40 ■

Such an automaton is said to be R/E-coherent. The intuition behind this is the following: in the tree automaton, $\Re$-transitions represent rewriting steps and transitions of $\mathcal{A}^{\not\Re}$ recognise E-equivalence classes. More precisely, in a R/E-coherent tree automaton, if two terms s, t are recognised into the same state q in $\mathcal{A}^{\not\Re}$ then they belong to the same E-equivalence class. Otherwise, if at least one $\Re$-transition is necessary to recognise, say, t into q then at least one step of rewriting was necessary to obtain t from s. In [GR10], the following theorem made an assumption of R/E-coherence for $\mathcal{A}_0$, but, given that $\mathcal{A}_0$ does not have any $\Re$-transition, $\mathcal{A}_0$ is R/E-coherent if and only if it is E-coherent.

Theorem 41 (Upper bound).
Let E be a set of equations, $\mathcal{A}_0$ a E-coherent tree automaton and R a left-linear TRS. If $\mathcal{A}$ is an automaton produced from $\mathcal{A}_0$ after several steps of equational completion with R, E, then

$$\mathscr{L}(\mathcal{A}) \subseteq (R/E)^*(\mathscr{L}(\mathcal{A}_0))$$ 41 ■

4. Adaptation to innermost strategies

4.1. Introduction

Our first contribution is an adaptation of the classical equational completion of [GR10]. The classical equational completion procedure with a left-linear TRS R produces a correct over-approximaton of $R^*(L_0)$ whenever it terminates. As $R^*_{\text{in}}(L_0) \subseteq R^*(L_0)$, this is a correct over-approximation of $R^*_{\text{in}}(L_0)$ as well. Still, we would like to refine this procedure to deal more precisely with R_{in}. Indeed, there are some critical pairs that we would not want to complete because they do not correspond to any *innermost* rewriting situation.

Example 42.
Let us look at Example 33. The rewriting of $f(c(a(s(0)), n))$ into $c(a(s(0)), f(n))$ does not conform to innermost strategy because $a(s(0))$ is not a normal form. We would like to abstain from completing CP_1 of Example 33. 42 ◁

Due to the definition of innermost rewriting, we will need to discriminate between normal forms and terms reducible by R. To do so is possible using the automaton $\mathcal{IRR}(R)$ (see Theorem 27). It is possible to build a product between $\mathcal{A}$ and $\mathcal{IRR}(R)$, the tree automaton recognising the normal forms of R. Let $\mathcal{A}_\circ$ be an automaton recognising the initial language. Completion will start with $\mathcal{A}_0 = \mathcal{A}_\circ \times \mathcal{IRR}(R)$. This automaton enjoys the following property, which we will be useful to prove correctness.

Definition 43 (Consistency with $\mathcal{IRR}(R)$).
A pair automaton $\mathcal{A}$ is said to be consistent with $\mathcal{IRR}(R)$ if, for any configuration c and any state $\langle q, p\rangle$ of $\mathcal{A}$, $\Pi_2(c)$ is a configuration and p is a state of $\mathcal{IRR}(R)$, and if $c \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} \langle q, p\rangle$ then $\Pi_2(c) \overset{*}{\underset{\mathcal{IRR}(R)}{\rightarrowtail}} p$. 43 ◀

In the next subsections, we will restate the definitions used by equational completion to adapt them to our new framework of pair automata. *The TRS R is always supposed left-linear.* Some parts of them might look tricky, and they are indeed tricks to preserve the property of consistency with $\mathcal{IRR}(R)$.

4.2. Exact completion

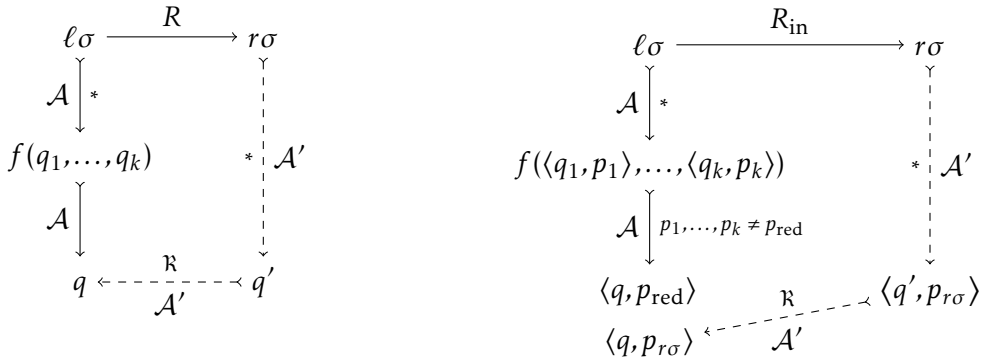
Definition 44 (Innermost critical pair).
A pair $(\ell \to r, \sigma, \langle q, p\rangle)$ where $\ell \to r \in R$, $\sigma : \mathcal{X} \to Q_\mathcal{A}$ and $\langle q, p\rangle \in Q_\mathcal{A}$ is critical if

1. $\ell\sigma \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} \langle q, p\rangle$,
2. there is no p' such that $r\sigma \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} \langle q, p'\rangle$ and
3. for each subconfiguration γ at depth 1 of $\ell\sigma$, the state $\left\langle q_\gamma, p_\gamma \right\rangle$ such that $\gamma \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} \left\langle q_\gamma, p_\gamma \right\rangle$ in the recognition path of condition 1 is with $p_\gamma \neq p_{\text{red}}$. 44 ◀

Remark. Because a critical pair denotes a rewriting situation, the p of definition 44 is necessarily p_{red} as long as $\mathcal{A}$ is consistent with $\mathcal{IRR}(R)$.

Example 45.
In the situation of Examples 17–28, consider the rule $f(c(a(X), Y)) \to c(a(X), f(Y))$, the substitution $\sigma_1 = \{X \mapsto \langle q_s, p_s\rangle, Y \mapsto \langle q_n, p_n\rangle\}$ and the state $\left\langle q_f, p_{\text{red}} \right\rangle$: this is not an innermost critical pair because the recognition path is $f(c(a(\langle q_s, p_s\rangle), \langle q_n, p_n\rangle)) \rightarrowtail f(c(\langle q_a, p_{\text{red}}\rangle, \langle q_n, p_n\rangle)) \rightarrowtail f(\langle q_c, p_{\text{red}}\rangle) \rightarrowtail \left\langle q_f, p_{\text{red}} \right\rangle$ and so there is a p_{red} at depth 1. But there is an innermost critical in $\mathcal{A}_0$ with the rule $a(s(X)) \to X$, the substitution $\sigma_2 = \{X \mapsto \langle q_0, p_0\rangle\}$ and the state $\langle q_a, p_{\text{red}}\rangle$. 45 ◁

Normalisation is like in the classical case, except we deal with pairs $\langle q, p\rangle$ instead of just q. Note that whenever we add a new transition $c' \rightarrowtail \langle q', p'\rangle$, only the q' is arbitrary: the p' is always the state of $\mathcal{IRR}(R)$ such that $projdroitec \overset{}{\underset{\mathcal{IRR}(R)}{\rightarrowtail}} p'$. Again, the formal definition presented in Appendix A, as it is technical and the details are not relevant to understanding the tree automata completion process.



(a) Classical (b) Innermost

Figure 2: Comparison of classical and innermost critical pairs

Example 46.

With a suitable signature, suppose that automaton $\mathcal{A}$ consists of the transitions $c \rightarrowtail \langle q_1, p_c\rangle$ and $f(\langle q_1,p_c\rangle) \rightarrowtail \left\langle q_2, p_{f(c)}\right\rangle$ and we want to normalise $f(g(\left\langle q_2,p_{f(c)}\right\rangle,c))$ to the new state $\left\langle q_N, p_{f(g(f(c),c))}\right\rangle$. We first have to normalise under g: $\left\langle q_2,p_{f(c)}\right\rangle$ is already a state, so it does not need to be normalised; c has to be normalised to a state: since $\mathcal{A}$ already has transition $c \rightarrowtail \langle q_1,p_c\rangle$, we add nothing and it remains to normalise $g(\left\langle q_2,p_{f(c)}\right\rangle,\langle q_1,p_c\rangle)$. Since $\mathcal{A}$ does not contain a transition for this configuration, we must add a new state $\left\langle q', p_{g(f(c),c)}\right\rangle$ and the transition $g(\left\langle q_2,p_{f(c)}\right\rangle,\langle q_1,p_c\rangle) \rightarrowtail \left\langle q', p_{g(f(c),c)}\right\rangle$. Finally, we add $f(\left\langle q', p_{g(f(c),c)}\right\rangle) \rightarrowtail \left\langle q_N, p_{f(g(f(c),c))}\right\rangle$. Note that whenever we add a new transition $c' \rightarrowtail \langle q',p'\rangle$, only the q' is arbitrary: the p' is always the state of $\mathcal{IRR}(R)$ such that $projdroite c \underset{\mathcal{IRR}(R)}{\rightarrowtail} p'$. 46 ◁

The following definition comprises two parts. The first set of operation formalises "closing the square"; the supplementary operations ensure that any context that was recognised after $\langle q,p_{\text{red}}\rangle$ will have a corresponding recognition path starting from $\langle q,p_{r\sigma}\rangle$.

Definition 47 (Completion of an innermost critical pair).

A critical pair $(\ell \to r,\sigma,\langle q,p\rangle)$ in automaton $\mathcal{A}$ is completed by first computing $N = Norm_{\mathcal{A}^{\mathcal{R}}}(r\sigma,\langle q',p_{r\sigma}\rangle)$ where q' is a new state and $\Pi_2(r\sigma) \overset{*}{\underset{\mathcal{IRR}(R)}{\rightarrowtail}} p_{r\sigma}$, then adding to $\mathcal{A}$ the new states and the transitions appearing in N as well as the transition $\langle q',p_{r\sigma}\rangle \overset{\mathcal{R}}{\rightarrowtail} \langle q,p_{r\sigma}\rangle$. If $r\sigma$ is a trivial configuration (ie. r is just a variable), only transition $r\sigma \overset{\mathcal{R}}{\rightarrowtail} \langle q,\Pi_2(r\sigma)\rangle$ is added.

Afterwards, we execute the following supplementary operations. For any transition

$$f(\ldots,\langle q,p_{\text{red}}\rangle,\ldots) \rightarrowtail \langle q'',p''\rangle,$$

we add (if it is not present) a transition

$$f(\ldots,\langle q,p_{r\sigma}\rangle,\ldots) \rightarrowtail \langle q'',p'''\rangle$$

with $f(\ldots,p_{r\sigma},\ldots) \underset{\mathcal{IRR}(R)}{\rightarrowtail} p'''$. These new transitions are in turn recursively considered for the supplementary operations.[2] 47 ◀

[2] We add no new state in these supplementary operations, so this necessarily terminates.

Definition 48 (Step of exact completion).
This is the same definition as Definition 32, but using Definition 47 instead of Definition 31 for each critical pair. 48 ◀

4.3. Equational simplification

Definition 49 (Situation of application of an equation).
Given an equation $s = t$, an automaton $\mathcal{A}$, a $\theta : \mathcal{X} \to Q_{\mathcal{A}}$ and states $\langle q_1, p_1\rangle$ and $\langle q_2, p_2\rangle$, we say that $(s = t, \theta, \langle q_1, p_1\rangle, \langle q_2, p_2\rangle)$ is a situation of application in $\mathcal{A}$ if

1. $s\theta \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} \langle q_1, p_1\rangle$,
2. $t\theta \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} \langle q_2, p_2\rangle$,
3. $\langle q_1, p_1\rangle \not\underset{\mathcal{A}}{\overset{E}{\rightarrowtail}} \langle q_2, p_2\rangle$ and
4. $p_1 = p_2$. 49 ◀

Definition 50 (Application of an equation).
Given $(s = t, \theta, \langle q_1, p_1\rangle, \langle q_2, p_1\rangle)$ a situation of application in $\mathcal{A}$, applying the underlying equation in it consists in adding transitions $\langle q_1, p_1\rangle \overset{E}{\rightarrowtail} \langle q_2, p_1\rangle$ and $\langle q_2, p_1\rangle \overset{E}{\rightarrowtail} \langle q_1, p_1\rangle$ to $\mathcal{A}$. We also add the supplementary transitions $\langle q_1, p_1'\rangle \overset{E}{\rightarrowtail} \langle q_2, p_1'\rangle$ and $\langle q_2, p_1'\rangle \overset{E}{\rightarrowtail} \langle q_1, p_1'\rangle$ where $\langle q_1, p_1'\rangle$ and $\langle q_2, p_1'\rangle$ are variants that occur in the automaton. 50 ◀

Lemma 51.
Applying an equation preserves consistency with $\mathcal{IRR}(R)$. 51 ■

Proof.
Let $\mathcal{A}$ be a consistent with $\mathcal{IRR}(R)$ automaton whose set of states is Q, let $\mathcal{B}$ result from the adjunction of transition $\langle q_1, p_1\rangle \rightarrowtail \langle q_2, p_1\rangle$ to $\mathcal{A}$ due to the application of some equation. Notice that this is sufficient because of the symmetry between q_1 and q_2. We proceed by induction on k, the number of times the transition $\langle q_1, p_1\rangle \rightarrowtail \langle q_2, p_1\rangle$ occurs in the path $c \underset{\mathcal{B}}{\overset{*}{\rightarrowtail}} \langle q, p\rangle$ where c is a configuration and $\langle q, p\rangle$ is a state of $\mathcal{B}$. If there is no occurrence, then $c \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} \langle q, p\rangle$ and by consistency of $\mathcal{A}$, $\Pi_2(c) \underset{\mathcal{IRR}(R)}{\overset{*}{\rightarrowtail}} p$.

Suppose the property is true for some k and there is a context C on $T(\Sigma, Q)$ such that $c \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} C[\langle q_1, p_1\rangle] \underset{\mathcal{B}}{\rightarrowtail} C[\langle q_2, p_1\rangle] \underset{\mathcal{B}}{\overset{*}{\rightarrowtail}} \langle q, p\rangle$ with the last part of the path using less than k times the new transition. First, there is a configuration c_1 such that $c = C[c_1]$ and $c_1 \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} \langle q_1, p_1\rangle$, and therefore $\Pi_2(c_1) \underset{\mathcal{IRR}(R)}{\overset{*}{\rightarrowtail}} p_1$. Second, by induction hypothesis, $\Pi_2(C[p_1]) \underset{\mathcal{IRR}(R)}{\overset{*}{\rightarrowtail}} p$. Finally, $\Pi_2(c) \underset{\mathcal{IRR}(R)}{\overset{*}{\rightarrowtail}} \Pi_2(C[p_1]) \underset{\mathcal{IRR}(R)}{\overset{*}{\rightarrowtail}} p$. □

Remark. The condition 4 ($p_1 = p_2$) of definition 49 is obviously necessary for this lemma. Here is why consistency is important and what would happen if we allowed equations to be applied without regard for condition 4 of Definition 49. Take $R = \{f(a) \to w, g(f(b)) \to c\}$, $E = \{a = b\}$ and $\mathcal{A}_\circ = \{a \rightarrowtail q_a, b \rightarrowtail q_b, f(q_a) \rightarrowtail q_{fa}, g(q_{fa}) \rightarrowtail q_{gfa}\}$. We have

$$\begin{aligned}\mathcal{IRR}(R) = \{&a \rightarrowtail p_a, b \rightarrowtail p_b, c \rightarrowtail p_c, w \rightarrowtail p_c,\\ &f(p_a) \rightarrowtail p_{\text{red}}, f(p_b) \rightarrowtail p_{fb}, f(p_c) \rightarrowtail p_c,\\ &g(p_{fb}) \rightarrowtail p_{\text{red}}, g(p_c) \rightarrowtail p_c\}\end{aligned}$$

We omit the transitions whose left-hand side contains p_{red}: they always have p_{red} as a right-hand side.

Remark that $c \in R_{\text{in}}(g(f(b)))$. However, $g(f(b))$ is not recognised by $\mathcal{A}_\circ$ into any state and thus not by $\mathcal{A}_\circ \times \mathcal{IRR}(R)$ either, so we are not interested in its successors. But let us apply the equation $a = b$ without taking condition 4 of definition 49 into account: in particular, we add the transition $\langle q_b, p_b\rangle \rightarrowtail \langle q_a, p_a\rangle$. Therefore we now have $g(f(b)) \rightarrowtail g(f(\langle q_b, p_b\rangle)) \rightarrowtail g(f(\langle q_a, p_a\rangle)) \rightarrowtail g(\left\langle q_{fa}, p_{\text{red}}\right\rangle) \rightarrowtail \left\langle q_{gfa}, p_{\text{red}}\right\rangle$. We have $g(f(b)) \in \mathcal{L}(\mathcal{A}, \left\langle q_{gfa}, p_{\text{red}}\right\rangle)$ and are thus now interested in its successors: we would like to have some critical pair involving $g(f(b))$ and c. But we do not, because the only recognition path of $g(f(b))$, the one we just created, does not fulfil condition 3 of definition 44.

5. Correctness

Lemma 52.

Let $\mathcal{A}$ be an automaton obtained from some $\mathcal{A}_\circ \times \mathcal{IRR}(R)$ after some steps of innermost completion. $\mathcal{A}$ is consistent with $\mathcal{IRR}(R)$. 52 ■

Proof.

$\mathcal{A}_\circ \times \mathcal{IRR}(R)$ is consistent with $\mathcal{IRR}(R)$ by construction. Lemma 51 shows that applying an equation preserves this. It remains to show that a step of exact completion does so as well.

The first steps of normalisation are preserving because the new $\left\langle q_i', p_i'\right\rangle$ are precisely chosen such that $\Pi_2\left(d_{|\xi_i}\right) \underset{\mathcal{IRR}(R)}{\overset{*}{\rightarrowtail}} p_i'$. The last step of normalisation is preserving too, as well as the remaining operations of the completion of a critical pair, because, again, we choose p' such that $r\sigma \underset{\mathcal{IRR}(R)}{\overset{*}{\rightarrowtail}} p'$.

The same goes for the supplementary operations. □

Lemma 53.

Let $\mathcal{A}$ be an automaton consistent with $\mathcal{IRR}(R)$, $(\ell \to r, \sigma, \langle q, p\rangle)$ a critical pair in $\mathcal{A}$, let $p_{r\sigma}$ be the state of $\mathcal{IRR}(R)$ such that $r\sigma \underset{\mathcal{IRR}(R)}{\overset{*}{\rightarrowtail}} p_{r\sigma}$ and $\mathcal{B}$ be the automaton resulting from the completion of this critical pair. Let C be a context on $T(\Sigma)$ and $\langle q_1, p_1\rangle$ a state of $\mathcal{A}$ such that $C[\langle q, p\rangle] \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} \langle q_1, p_1\rangle$. Then there exists a state p_2 of $\mathcal{IRR}(R)$ such that $C[\langle q, p_{r\sigma}\rangle] \underset{\mathcal{B}}{\overset{*}{\rightarrowtail}} \langle q_1, p_2\rangle$. 53 ■

Proof.

Note that we have $p = p_1 = p_{\text{red}}$. We have to show that all the transitions used in the path $C[\langle q, p_{\text{red}}\rangle] \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} \langle q_1, p_{\text{red}}\rangle$ have some counterpart starting from $C[\langle q, p_{r\sigma}\rangle]$. First, observe that all

transitions used to recognise subterms at positions of C that are parallel to the position of the hole can remain unchanged. It remains to be shown that the transitions involving positions in the branch of the completion of the considered critical pair also have their counterpart: they exist thanks to the supplementary operations of definition 47. □

Remark. The supplementary operations are necessary for this lemma. Indeed, take $R = \{g(f(b)) \to g(f(a)), f(a) \to c\}$, $\mathcal{A}_\circ = \{b \rightarrowtail q_b, f(q_b) \rightarrowtail q_{fb}, g(q_{fb}) \rightarrowtail q_{gfb}\}$. We have $\mathcal{IRR}(R) = \{a \rightarrowtail p_a, b \rightarrowtail p_b, c \rightarrowtail p_c, f(p_a) \rightarrowtail p_{\text{red}}, g(p_a) \rightarrowtail p_c, f(p_b) \rightarrowtail p_{fb}, g(p_{fb}) \rightarrowtail p_{\text{red}}, f(p_c) \rightarrowtail p_c, g(p_c) \rightarrowtail p_c\}$. There is a critical pair $PC_1 = (g(f(b)) \to g(f(a)), \varnothing, \left\langle q_{gfb}, p_{\text{red}} \right\rangle)$ in $\mathcal{A}_\circ \times \mathcal{IRR}(R)$, which is resolved by adding transitions $a \rightarrowtail \langle q_{N1}, p_a\rangle$, $f(\langle q_{N1}, p_a\rangle) \rightarrowtail \langle q_{N2}, p_{\text{red}}\rangle$, $g(\langle q_{N2}, p_{\text{red}}\rangle) \rightarrowtail \langle q_{N3}, p_{\text{red}}\rangle$ and $\langle q_{N3}, p_{\text{red}}\rangle \rightarrowtail \left\langle q_{gfb}, p_{\text{red}} \right\rangle$, thereby producing automaton $\mathcal{A}_1$. The supplementary operations do not create any new transition here.

There is a critical pair $PC_2 = (f(a) \to c, \varnothing, \langle q_{N2}, p_{\text{red}}\rangle)$ in $\mathcal{A}_1$, which is resolved by adding transitions $c \rightarrowtail \langle q_{N4}, p_c\rangle$ and $\langle q_{N2}, p_c\rangle$, thereby producing automaton $\mathcal{A}_2^\natural$. The supplementary operations are detailed further down and produce automaton $\mathcal{A}_2$.

Now consider that $g(c) \in R_{\text{in}}(g(f(a)))$ and $g(f(a)) \in \mathscr{L}(\mathcal{A}_2^\natural, \left\langle q_{gfb}, p_{\text{red}} \right\rangle)$ because we completed PC_1. But all what we have is $g(c) \underset{\mathcal{A}_2^\natural}{\rightarrowtail} g(\langle q_{N4}, p_c\rangle) \underset{\mathcal{A}_2^\natural}{\rightarrowtail} g(\langle q_{N2}, p_c\rangle)$, this last configuration being not productive. As a result, $g(c) \notin \mathscr{L}(\mathcal{A}_2^\natural, \left\langle q_{gfb}, p' \right\rangle)$ for any p'.

The supplementary operations are made after completion of PC_2. Since there is a transition $g(\langle q_{N2}, p_{\text{red}}\rangle) \rightarrowtail \langle q_{N3}, p_{\text{red}}\rangle$, we add a transition $g(\langle q_{N2}, p_c\rangle) \rightarrowtail \langle q_{N3}, p_c\rangle$. Then, since $q_{N3} \notin \mathcal{A}_\circ$ either, and there is a $\langle q_{N3}, p_{\text{red}}\rangle \rightarrowtail \left\langle q_{gfb}, p_{\text{red}} \right\rangle$, we add $\langle q_{N3}, p_c\rangle \rightarrowtail \left\langle q_{gfb}, p_c \right\rangle$. No further transition needs to be added. These transitions allow $g(c) \in \mathscr{L}(\mathcal{A}_2, \left\langle q_{gfb}, p_c \right\rangle)$.

Definition 54 (Correct automaton).
An automaton $\mathcal{A}$ is correct wrt. R_{in} if for all state $\langle q, p_{\text{red}}\rangle$ of $\mathcal{A}$, for all $u \in \mathscr{L}(\mathcal{A}, \langle q, p_{\text{red}}\rangle)$ and for all $v \in R_{\text{in}}(u)$, either there is a state p of $\mathcal{IRR}(R)$ such that $v \in \mathscr{L}(\mathcal{A}, \langle q, p\rangle)$ or there is a critical pair $(\ell \to r, \sigma, \langle q_0, p_0\rangle)$ in $\mathcal{A}$ for some $\langle q_0, p_0\rangle$ and a context C on $T(\Sigma)$ such that $u \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} C[\ell\sigma] \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} C[\langle q_0, p_{\text{red}}\rangle] \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} \langle q, p_{\text{red}}\rangle$ and $v \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} C[r\sigma]$. 54 ◀

Lemma 55.
Any automaton produced by innermost completion starting from some $\mathcal{A}_\circ \times \mathcal{IRR}(R)$ is correct wrt. R_{in}. 55 ■

Proof.
Let $\mathcal{A}$ be such an automaton; it is consistent with $\mathcal{IRR}(R)$. Let $\langle q, p_{\text{red}}\rangle$ be a state of $\mathcal{A}$, $u \in \mathscr{L}(\mathcal{A}, \langle q, p_{\text{red}}\rangle)$ and $v \in R_{\text{in}}(u)$. By definition of innermost rewriting, there is a rule $\ell \to r$ of R, a substitution $\mu : \mathcal{X} \to T(\Sigma)$ and a context C such that $u = C[\ell\mu]$, $v = C[r\mu]$ and each strict subterm of u is a normal form. Let $u_0 = \ell\mu$ and $v_0 = r\mu$. There is a $\langle q_0, p_{\text{red}}\rangle$ such that $u_0 \in \mathscr{L}(\mathcal{A}, \langle q_0, p_{\text{red}}\rangle)$ and $C[\langle q_0, p_{\text{red}}\rangle] \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} \langle q, p_{\text{red}}\rangle$.

Since ℓ is linear, there is a $\sigma : \mathcal{X} \to Q_\mathcal{A}$ such that $\ell\mu \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} \ell\sigma \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} \langle q_0, p_{\text{red}}\rangle$ and $r\mu \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} r\sigma$. This entails that $u \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} C[\ell\sigma] \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} C[\langle q_0, p_{\text{red}}\rangle] \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} \langle q, p_{\text{red}}\rangle$ and $v \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} C[r\sigma]$.

Assume that there is no p_0 such that $v_0 \in \mathscr{L}(\mathcal{A}, \langle q_0, p_0\rangle)$ and show that $(\ell \to r, \sigma, \langle q_0, p_{\text{red}}\rangle)$ is a critical pair in $\mathcal{A}$. First, by assumption, there is no p such that $r\sigma \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} \langle q_0, p\rangle$. Conditions 1

and 2 of definition 44 are thus met. Suppose $\ell = f(\gamma_1, \ldots, \gamma_k)$ and show that condition 3 of definition 44 holds.[3] For each $i \in [\![1\,;k]\!]$, let $\langle q_i, p_i\rangle$ be the state of $\mathcal{A}$ such that $\gamma_i\mu \xrightarrow[\mathcal{A}]{*} \gamma_i\sigma \xrightarrow[\mathcal{A}]{*} \langle q_i, p_i\rangle$ in the path of recognition of $\ell\sigma$. Then, by consistency with $\mathcal{IRR}(R)$, for each $i \in [\![1\,;k]\!]$, $\gamma_i\mu \xrightarrow[\mathcal{IRR}(R)]{*} p_i$. Since strict subterms of $\ell\mu$ are strict subterms of u as well, they are normal forms, thus $p_i \neq p_{\text{red}}$, which validates condition 3 of definition 44.

Assume now that $v_0 \in \mathscr{L}(\mathcal{A}, \langle q_0, p_0\rangle)$ and show that there is a p such that $v \in \mathscr{L}(\mathcal{A}, \langle q, p\rangle)$. This is obvious at the initial step $\mathcal{A}_\circ \times \mathcal{IRR}(R)$, and this property is conserved by completion as shown by lemma 53. □

Theorem 56 (Correctness).

Assuming R is left-linear, the innermost equational completion procedure defined above produces a correct result whenever it terminates and produces some fixpoint $\mathcal{A}_{\text{in}*}$:

$$\mathscr{L}(\mathcal{A}_{\text{in}*}) \supseteq R^*_{\text{in}}(\mathscr{L}(\mathcal{A}_\circ \times \mathcal{IRR}(R))). \qquad 56\ \blacksquare$$

Proof.

Let $\mathcal{A}_{\text{in}*}$ be the calculated fixpoint automaton. By lemma 52, $\mathcal{A}_{\text{in}*}$ is consistent with $\mathcal{IRR}(R)$, and therefore, by lemma 55, $\mathcal{A}_{\text{in}*}$ is correct wrt. R_{in}. Since this automaton is a fixpoint, the case of definition 54 where there remains a critical pair cannot occur, and therefore, for all state $\langle q, p_{\text{red}}\rangle$ of $\mathcal{A}$, for all $u \in \mathscr{L}(\mathcal{A}_{\text{in}*}, \langle q, p_{\text{red}}\rangle)$ and for all $v \in R_{\text{in}}(u)$, there is a p' such that $v \in \mathscr{L}(\mathcal{A}_{\text{in}*}, \langle q, p\rangle)$. Thus, $\mathcal{A}_{\text{in}*}$ is R_{in}-closed: $R_{\text{in}}(\mathscr{L}(\mathcal{A}_{\text{in}*})) \subseteq \mathscr{L}(\mathcal{A}_{\text{in}*})$. Since the completion process only adds transitions, $\mathscr{L}(\mathcal{A}_{\text{in}*}) \supseteq \mathscr{L}(\mathcal{A}_\circ \times \mathcal{IRR}(R))$, which concludes the proof. □

6. Using innermost completion to improve accuracy of static analysis

We made an alpha implementation of innermost completion. This new version of Timbuk, named TimbukSTRAT, is available at [Tim14] along with several examples. On those examples, innermost completion run within milliseconds. Sets of approximation equations, when needed, are systematically defined [GS13].

We now explain two of those examples `add` and `sumlist1` showing that this tool can improve the accuracy of static analysis techniques. TRSs, and similar formalisms, have been shown to be a convenient model for static analysis of functional programs [Jon87, KO11] and Java programs [BGJL07]. In this setting, the static analysis tool itself only builds approximations of terms reachable with the TRS encoding the semantics of the program.

6.1. More relevant analysis for TRSs encoding the Java semantics

In the Java semantics, the `add` bytecode performs an addition of two integers x and y of a stack. Here is a simplified version of what should be the TRS encoding the effect of the `add` Java bytecode on a stack of integers:

$plus(0, X) \to X$
$plus(s(X), Y) \to s(plus(X, Y))$
$add(stack(X, stack(Y, Z))) \to stack(plus(X, Y), Z)$

[3] If ℓ is a constant, then condition 3 is vacuously true.

However, to be coherent with Java semantics, this TRS should ensure that the addition x+y has been normalized before storing its result on the same stack. Since the tools building the approximations of reachable terms are not strategy-aware, they build rough approximations containing stacks of non-reduced integer additions, which are not reachable w.r.t. the Java semantics. In [BGJL07], the TRS representing the semantics of a Java programs had to be transformed so as to encode, in the TRS, the call-by-value of Java. Furthermore, a similar transformation is necessary for many of the Java bytecode operations, leading to an unnecessarily complicated TRS. This is also the case with methods commonly used to encode the innermost strategy into the program itself, such as continuation-passing-style transformations [Plo75, DF92].

Using innermost completion can totally avoid this problem. On the above TRS, the set of descendants computed by Timbuk STRAT with file `add` does not contain stacks of non-reduced integer additions.

6.2. Improving static analysis of functional programs

There is a recent and renewed interest for Data flow analysis of higher-order functional programs [OR11, KO11, KI13] that was initiated by [Jon87]. On examples taken from [OR11], we showed in [GS13] that completion can compete with those techniques. None of those techniques is strategy-aware: on Example 17, they all consider the term $c(a(s(0)), f(n))$ as reachable, though it is not with innermost strategy. Example 17 also shows that this is not the case with innermost completion.

Accuracy of innermost completion can benefit to static analysis of functional programs. To illustrate this, we choose to show that approximations built with TimbukSTRAT are precise enough to prove non-termination of functions using call-by-value evaluation strategy. First, we recall how to prove non-termination using an approximation of reachable terms [Gen98]. Consider the following OCaml program, which can be represented as a TRS R:[4]

```
1 let hd= function x::_ -> x;;
2 let tl= function _::l -> l;;
3 let rec delete(e,l)=
4  if (l=[]) then []
5  else if (hd l=e) then tl l else (hd l)::delete(e,l);;
```

It is faulty: the recursive call should be `(hd l)::delete(e, tl l)`. Any call `delete(e, l)` will not terminate if `l` is not empty and `hd l` is not `e`. Since the language of such calls to `delete` with `l` not empty is regular, completion can compute an automaton over-approximating of the set of reachable terms from this language. Besides, we can compute the automaton $\mathcal{IRR}(R)$ recognizing normal forms of R[5]. Then, computing the intersection between the two automata, we obtain the automaton recognizing the set of reachable terms in normal form[6]: this set is empty. This reflects the fact that the `delete` function does not compute any *result*, i.e. it is not terminating.

Now, we perform the same kind of non-termination proof on a program which is terminating with call-by-name but not with call-by-value strategy.[7] The OCaml program `sum x` computes

[4] See file `nonTerm1` under TimbukSTRAT at [Tim14].

[5] For functional programs, computing $\mathcal{IRR}(R)$ can be easier than for general TRS: it can consist of the simple automaton recognising constructor terms [GS13].

[6] Computing $\mathcal{IRR}(R)$ and the intersection can be done using Timbuk.

[7] See file sumlist1.txt.

the sum of the x first natural numbers. It does not terminate because of OCaml's call-by-value evaluation. Strategy-unaware methods cannot show this: there are (outermost) reachable terms that are in normal forms: the integer results obtained with a call-by-need or lazy evaluation.

```
let rec sumList(x,y)=
  (x+y)::sumList(x+y,y+1);;
let rec nth i (x::l)=
  if i<=0 then x else nth (i-1) l;;
let sum x= nth x (sumList(0,0));;
```

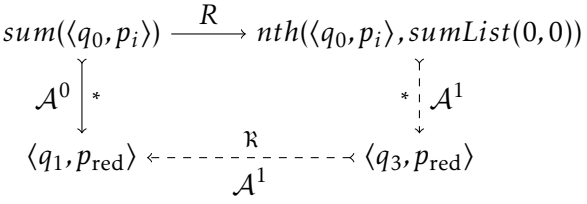
(1) $0 + X \to X$
(2) $s(X) + Y \to s(X + Y)$
(3) $sumList(X, Y) \to cons(X + Y, sumList(X + Y, s(Y)))$
(4) $nth(0, cons(X, Y)) \to X$
(5) $nth(s(X), cons(Y, Z)) \to nth(X, Z)$
(6) $sum(X) \to nth(X, sumList(0, 0))$

```
Ops sum:1 nth:2 sumList:2 cons:2 nil:0 zero:0 s:1 add:2
Vars X Y Z U

TRS R1
add(zero,X) -> X
add(s(X),Y) -> s(add(X,Y))
sumList(X,Y) -> cons(add(X,Y), sumList(add(X,Y),s(Y)))
nth(zero,cons(X,Y)) -> X
nth(s(X),cons(Y,Z)) -> nth(X,Z)
sum(X) -> nth(X,sumList(zero,zero))
```

Assume that we want to over-approximate the set of results of the function `sum` for all natural number `i`. Thus, our aim is to compute the automaton recognising the set of *innermost* reachable terms from the initial regular language $\{sum(s^*(0))\}$. Let $\mathcal{A} = (\Sigma, Q, Q_f, \Delta)$ with $Q_f = \{q_1\}$ and $\Delta = \{0 \rightarrowtail q_0, s(q_0) \rightarrowtail q_0, sum(q0) \rightarrowtail q_1\}$ be an automaton recognising this language. Timbuk [Tim14] can compute the automaton $\mathcal{IRR}(R)$, whose description is then appended to the definition of the TRS in `sumlist1.txt`. In this automaton, all transitions with top symbol *sum* are mapped to p_{red}. This is not surprising since the rule $sum(X) \to nth(X, sumList(0, 0))$ rewrites any term rooted by *sumList*, *i.e.* they are all reducible. This is the same for all transitions rooted by the *sumList* symbol.

On the opposite, terms built only on symbols s and 0 are all irreducible. Let us denote by p_i the state recognising them in $\mathcal{IRR}(R)$. Let us denote by $\mathcal{A}_0, \mathcal{A}_1, \ldots$ the tree automata obtained by innermost completion. Let $\mathcal{A}_0 = \mathcal{A}_\circ \times \mathcal{IRR}(R)$ be the initial product automaton whose set of transitions is: $0 \rightarrowtail \langle q_0, p_i\rangle$, $s(\langle q_0, p_i\rangle) \rightarrowtail \langle q_0, p_i\rangle$, $sum(\langle q_0, p_i\rangle) \rightarrowtail \langle q_1, p_{\text{red}}\rangle$. The first completion step solves only one critical pair which is:



During the normalisation of the new transition $nth(\langle q_0, p_i\rangle, sumList(0, 0)) \rightarrowtail \langle q_3, p_{\text{red}}\rangle$, the subterm 0 is normalised using the transition $0 \rightarrowtail \langle q_0, p_i\rangle$ of $\mathcal{A}_0$ and the subterm $sumList(\langle q_0, p_i\rangle, \langle q_0, p_i\rangle)$ is normalised using a new state $\langle q_2, p_{\text{red}}\rangle$. Thus, according to Definition 47, the set of transitions added to $\mathcal{A}_0$ to obtain $\mathcal{A}_1$ is $nth(\langle q_0, p_i\rangle, \langle q_2, p_{\text{red}}\rangle) \rightarrowtail \langle q_3, p_{\text{red}}\rangle$, $sumList(\langle q_0, p_i\rangle, \langle q_0, p_i\rangle) \rightarrowtail \langle q_2, p_{\text{red}}\rangle$, and $\langle q_3, p_{\text{red}}\rangle \overset{\mathcal{R}}{\rightarrowtail} \langle q_1, p_{\text{red}}\rangle$. Since no other critical pair with symbol *sum* will never be solved, the language of reachable terms from $sum(s^*(0))$ is recognised by q_1 and thus by q_3. For q_3 to recognise *irreducible* terms, it necessary to discard the *nth* symbol by applying either rule (4) or (5) of R, *i.e.* have an innermost critical pair between rules (4) or (5) and the

transition $nth(\langle q_0, p_i\rangle, \langle q_2, p_{\text{red}}\rangle) \rightarrowtail \langle q_3, p_{\text{red}}\rangle$. However, this cannot happen because q_2 does not recognise any irreducible term. Furthermore, during all the following completion steps, terms recognised by q_2 will all be reducible. We can illustrate this on the second completion step which only solves one critical pair:

$$\begin{array}{ccc} sumList(\langle q_0, p_i\rangle, \langle q_0, p_i\rangle) & \xrightarrow{R} & cons(\langle q_0, p_i\rangle + \langle q_0, p_i\rangle, sumList(\langle q_0, p_i\rangle + \langle q_0, p_i\rangle, s(\langle q_0, p_i\rangle))) \\ \mathcal{A}^1 \downarrow * & & * \downarrow \mathcal{A}^2 \\ \langle q_2, p_{\text{red}}\rangle & \overset{\mathfrak{K}}{\underset{\mathcal{A}^2}{\dashleftarrow}} & \langle q_6, p_{\text{red}}\rangle \end{array}$$

Normalisation of the corresponding transition results in the following set of new transitions to be added to $\mathcal{A}_1$: $cons(\langle q_4, p_{\text{red}}\rangle, \langle q_5, p_{\text{red}}\rangle) \rightarrowtail \langle q_6, p_{\text{red}}\rangle$, $\langle q_0, p_i\rangle + \langle q_0, p_i\rangle \rightarrowtail \langle q_4, p_{\text{red}}\rangle$, $sumList(\langle q_4, p_{\text{red}}\rangle, \langle q_0, p_i\rangle) \rightarrowtail \langle q_5, p_{\text{red}}\rangle$ and $\langle q_6, p_{\text{red}}\rangle \overset{\mathfrak{K}}{\rightarrowtail} \langle q_2, p_{\text{red}}\rangle$. Note that right-hand side of this critical pair is recognised by the state $\langle q_6, p_{\text{red}}\rangle$. In fact, as explained above, all terms rooted by a *sumList* symbol will necessarily be recognised by p_{red}. As a consequence, we will never be able to perform an innermost completion step on the transition $nth(\langle q_0, p_i\rangle, \langle q_2, p_{\text{red}}\rangle) \rightarrowtail \langle q_3, p_{\text{red}}\rangle$ because the state $\langle q_2, p_{\text{red}}\rangle$ and all its successors will only contain terms having *sumList* symbols and thus being reducible. Using equations, it is possible to guarantee the termination of the completion (see [GS13]). On this example, having a set of equations containing at least equations $s(s(X)) = s(X)$ and $add(0, X) = X$ is enough to have a terminating completion. Rule (3) produces infinite lists with infinite sums of the form $(((0+1)+3)+6)+\ldots$. The equations are used to merge together all the integers. Completion with TimbukSTRAT terminates on an automaton (see section B) where the only product state labelled by q_1 is $\langle q_1, p_{\text{red}}\rangle$. This means that terms of the form $sum(s^*(0))$ have no innermost normal form. Thus, this proves that the function `sum` is *not terminating* in OCaml, which uses a call-by-value strategy.

On the same example, all aforementioned techniques [OR11, KO11, KI13, Jon87], as well as all completion techniques [Tak04, GR10, Lis12], give a more coarse approximation and are unable to prove non-termination with call-by-value. Indeed, those techniques approximate all reachable terms, independently of the rewriting strategy. Their approximation will, in particular, contain the integer results that are reachable by a call-by-need evaluation strategy.

7. Precision theorem

We showed that accuracy of the approximation has a great impact on the quality of the static analysis. Now we formally quantify this accuracy w.r.t. rewriting modulo E. For this, we need the hypothesis that the initial automaton does not mix different equivalence classes into the same state.

Definition 57 (Separation of E-classes).
The pair automaton $\mathcal{A}$ separates the classes of E if for any $q \in \Pi_1(Q_{\mathcal{A}})$, there is a term s such that for all $p \in \Pi_2(Q_{\mathcal{A}})$, $\mathscr{L}(\mathcal{A}, \langle q, p\rangle) \subseteq [s]_E$. 57 ◀

The following definitions are internal to the proof. We first give a notation for the common class of the terms recognised into a given state and extend this for configurations of the automaton.

Definition 58 (Equivalence class of a configuration).
Let $\mathcal{A}$ be an automaton that separates the classes of E. Let q be a left-component of a state ($q \in \Pi_1(\mathcal{A})$). If there is a $p \in \Pi_2(\mathcal{A})$ such that $\mathscr{L}(\mathcal{A}, \langle q, p\rangle) \neq \varnothing$, then the common E-class of all terms recognised by $\mathcal{A}$ into states of the form $\langle q, \cdot\rangle$ is noted $[q]_E^{\mathcal{A}}$. If there is no such p, we note $[q]_E^{\mathcal{A}} = \bot$.

We extend this notation to configurations of $\mathcal{A}$ as follows. First, we note $[\langle q, p\rangle]_E^{\mathcal{A}} = [q]_E^{\mathcal{A}}$. Then, let $c = f(c_1, \ldots, c_k)$ be a configuration of $\mathcal{A}$. If there is an $i \in [\![1\,;k]\!]$ such that $[c_i]_E^{\mathcal{A}} = \bot$, then let $[c]_E^{\mathcal{A}} = \bot$. Else, for each $i \in [\![1\,;k]\!]$, let $t_i \in \mathscr{L}(\Pi_1(\mathcal{A}), \Pi_1(c_i))$. We note $[c]_E^{\mathcal{A}} = [f(t_1, \ldots, t_n)]_E$. 58 ◀

Definition 59 (Total separation of classes).
Let $\mathcal{A}$ be an automaton separating the classes of E. We say that this separation is total if for any configuration c of $\mathcal{A}$, $[c]_E^{\mathcal{A}} \neq \bot$. 59 ◀

Now, some purely technical remarks to make the proof more readable.

Definition 60 (Reflexivity equations).
For a signature $(\Sigma, \mathcal{X})$, the set of its reflexivity equations is

$$E_r = \{f(x_1, \ldots, x_k) = f(x_1, \ldots, x_k) \mid k \in \mathbb{N} \wedge f \in \Sigma_k \wedge x_1, \ldots, x_k \text{ are distinct}\}.$$ 60 ◀

Remark. To ease the formulation of the proof, we will use a normalisation procedure that does not reuse already existing states, thus always normalising to new states. To compensate for this, we will add to E the set E_r of reflexivity equations: equational simplification will merge the superfluous states that were created by this modified normalisation procedure. On the other hand, the equivalence classes of $\equiv_E$ are the same as those of $\equiv_{E \cup E_r}$ and equational simplification makes recognised languages bigger, so this way of doing is valid.

Here follows the property that will be passed on to each completed automaton $\mathcal{A}_1$, etc.

Definition 61 (R_{in}/E-coherence).
An automaton $\mathcal{A}$ is R_{in}/E-coherent if

1. $\mathcal{A}^{\not K}$ totally separates the classes of E,
2. $\mathcal{A}$ is accessible, and
3. for any state $\langle q, p\rangle$ of $\mathcal{A}$, $\mathscr{L}(\mathcal{A}, \langle q, p\rangle) \subseteq (R_{\text{in}}/E)^*\left([q]_E^{\mathcal{A}^{\not K}}\right)$. 61 ◀

The following two simple lemmas show that we do equational simplification or completion "for a reason".

Lemma 62.
Let $\mathcal{A}$ be an automaton that totally separates the classes of E. Let $(s = t, \theta, \langle q_1, p\rangle, \langle q_2, p\rangle)$ be a situation of application of an equation of E in $\mathcal{A}$. Then $[q_1]_E^{\mathcal{A}} = [q_2]_E^{\mathcal{A}}$. 62 ■

Proof.
It suffices to prove that $[s\theta]_E^{\mathcal{A}} = [t\theta]_E^{\mathcal{A}}$. Since the separation of the classes by $\mathcal{A}$ is total, for each x in the domain of θ, there is a term $x\mu \in \mathscr{L}(\Pi_1(\mathcal{A}), x\theta)$. This builds an instance $s\mu \equiv_E t\mu$ of the considered equation. But $[s\theta]_E^{\mathcal{A}} = [s\mu]_E$ and $[t\theta]_E^{\mathcal{A}} = [t\mu]_E$. □

Lemma 63.
Let $\mathcal{A}$ be a R_{in}/E-coherent automaton that is consistent with $\mathcal{IRR}(R)$, let $(\ell \to r, \sigma, \langle q, p_{\text{red}}\rangle)$ be a critical pair that is to be completed by adding transition $\langle q', p_{r\sigma}\rangle \overset{\mathfrak{R}}{\rightarrowtail} \langle q, p_{r\sigma}\rangle$. We suppose that the normalisation steps have just been performed and still note $\mathcal{A}$ the resulting automaton. We have $\mathscr{L}(\mathcal{A}, r\sigma) \subseteq (R_{\text{in}}/E)^*\left([q]_E^{\mathcal{A}^{\not\mathfrak{R}}}\right)$. 63 ■

Proof.
let $t \in T(\Sigma)$ such that $t \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} r\sigma \overset{*}{\underset{\mathcal{A}^{\not\mathfrak{R}}}{\rightarrowtail}} \langle q', p_{r\sigma}\rangle$. For each variable x of r, let $x\mu$ be the subterm of t at the position where x occurs in r: we have thus $t = r\mu$. For each variable y appearing in ℓ but not in r, let $y\mu$ be any term in $\mathscr{L}(\mathcal{A}, y\sigma)$. Since $\mathcal{A}$ is consistent with $\mathcal{IRR}(R)$ and the critical pair fulfils Condition 3 of Definition 44, each strict subterm of $\ell\mu$ is a normal form. So $t = r\mu \in R_{\text{in}}(\ell\mu)$. Moreover, $\ell\mu \in \mathscr{L}(\mathcal{A}, \langle q, p_{\text{red}}\rangle)$ and $\mathcal{A}$ is R_{in}/E-coherent, so $t \in (R_{\text{in}}/E)^*\left([q]_E^{\mathcal{A}^{\not\mathfrak{R}}}\right)$. □

This is the main theorem of preservation.

Theorem 64.
Equational simplification preserves R_{in}/E-coherence. 64 ■

Proof.
Let $(s = t, \theta, \langle q_1, p_0\rangle, \langle q_2, p_0\rangle)$ be a situation of application of an equation of E in $\mathcal{A}$ and let $\mathcal{B}$ be the automaton resulting from the merger of $\langle q1, p_0\rangle$ and $\langle q_2, p_0\rangle$. Let $\langle q, p\rangle$ be a state.

Show that $\mathscr{L}(\mathcal{B}^{\not\mathfrak{R}}, \langle q, p\rangle) \subseteq [q]_E^{\mathcal{A}^{\not\mathfrak{R}}}$. Consider $u = C[u_0] \overset{*}{\underset{\mathcal{A}^{\not\mathfrak{R}}}{\rightarrowtail}} C[\langle q_1, p_0\rangle] \overset{E}{\rightarrowtail} C[\langle q_2, p_0\rangle] \overset{*}{\underset{\mathcal{A}^{\not\mathfrak{R}}}{\rightarrowtail}} \langle q, p\rangle$. We have $u_0 \in [q_1]_E^{\mathcal{A}^{\not\mathfrak{R}}}$, thus, by Lemma 62, $u_0 \in [q_2]_E^{\mathcal{A}^{\not\mathfrak{R}}}$. Therefore $u \in C[[q_2]_E^{\mathcal{A}^{\not\mathfrak{R}}}]$, which is just $[q]_E^{\mathcal{A}^{\not\mathfrak{R}}}$. Other cases are either trivial, symmetrical or reducible to this one.

Next, show that $\mathscr{L}(\mathcal{B}, \langle q, p\rangle) \subseteq (R_{\text{in}}/E)^*\left([q]_E^{\mathcal{A}^{\not\mathfrak{R}}}\right)$. Consider $u = C[u_0] \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} C[\langle q_1, p_0\rangle] \overset{E}{\rightarrowtail} C[\langle q_2, p_0\rangle] \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} \langle q, p\rangle$. We have $u_0 \in (R_{\text{in}}/E)^*\left([q_1]_E^{\mathcal{A}^{\not\mathfrak{R}}}\right)$, i.e. $u_0 \in (R_{\text{in}}/E)^*\left([q_2]_E^{\mathcal{A}^{\not\mathfrak{R}}}\right)$. Thus $u \in (R_{\text{in}}/E)^*\left(C[[q_2]_E^{\mathcal{A}^{\not\mathfrak{R}}}]\right)$.

First, assume that $C[\langle q_2, p_0\rangle] \overset{*}{\underset{\mathcal{A}^{\not\mathfrak{R}}}{\rightarrowtail}} \langle q, p\rangle$. Then $C[[q_2]_E^{\mathcal{A}^{\not\mathfrak{R}}}] = [q]_E^{\mathcal{A}^{\not\mathfrak{R}}}$, therefore $u \in (R_{\text{in}}/E)^*\left([q]_E^{\mathcal{A}^{\not\mathfrak{R}}}\right)$.

Second, assume that $C[\langle q_2, p_0\rangle] \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} \langle q, p\rangle$ with just one $\mathfrak{R}$-transition, that is $C[\langle q_2, p_0\rangle] \overset{*}{\underset{\mathcal{A}^{\not\mathfrak{R}}}{\rightarrowtail}} D[r\sigma] \overset{*}{\underset{\mathcal{A}^{\not\mathfrak{R}}}{\rightarrowtail}} D[\langle q_3', p_3\rangle] \overset{\mathfrak{R}}{\rightarrowtail} D[\langle q_3, p_3\rangle] \overset{*}{\underset{\mathcal{A}^{\not\mathfrak{R}}}{\rightarrowtail}} \langle q, p\rangle$. There is a corresponding critical pair $(\ell \to r, \sigma, \langle q_3, p_{\text{red}}\rangle)$ and, by Lemma 63, $\mathscr{L}(\mathcal{A}, r\sigma) \subseteq (R_{\text{in}}/E)^*\left([q_3]_E^{\mathcal{A}^{\not\mathfrak{R}}}\right)$. On the other hand, $D[[q_3]_E^{\mathcal{A}^{\not\mathfrak{R}}}] = [q]_E^{\mathcal{A}^{\not\mathfrak{R}}}$, so $\mathscr{L}(\mathcal{A}, D[r\sigma]) \subseteq (R_{\text{in}}/E)^*\left([q]_E^{\mathcal{A}^{\not\mathfrak{R}}}\right)$. Since $(R_{\text{in}}/E)^*$ is an operator that deals with equivalence classes, every term equivalent to one of $\mathscr{L}(\mathcal{A}, D[r\sigma])$ is also a descendant of $[q]_E^{\mathcal{A}^{\not\mathfrak{R}}}$ by R_{in}/E. Since $C[[q_2]_E^{\mathcal{A}^{\not\mathfrak{R}}}] = D[[r\sigma]_E^{\mathcal{A}^{\not\mathfrak{R}}}]$, u is a descendant of such a term, so $u \in (R_{\text{in}}/E)^*\left([q]_E^{\mathcal{A}^{\not\mathfrak{R}}}\right)$.

Finally, in the paragraph above, it suffices that $D[[q_3]_E^{\mathcal{A}^{\not\mathfrak{R}}}] \subseteq (R_{\text{in}}/E)^*\left([q]_E^{\mathcal{A}^{\not\mathfrak{R}}}\right)$ (we had $D[[q_3]_E^{\mathcal{A}^{\not\mathfrak{R}}}] = [q]_E^{\mathcal{A}^{\not\mathfrak{R}}}$): this allows us to reuse this case as an induction step over the number of $\mathfrak{R}$-transitions present in the path $C[\langle q_2, p_0\rangle] \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} \langle q, p\rangle$. □

Because we decided to always use fresh states for normalisation, the step about completion of a critical pair is simpler.

Lemma 65.
Normalisation preserves R_{in}/E-coherence. So do the supplementary operations performed after completing a critical pair. 65 ■

Lemma 66.
Completion of a critical pair preserves R_{in}/E-coherence. 66 ■

Proof.
Let $(\ell \to r, \sigma, \langle q, p_{\text{red}}\rangle)$ be a critical pair and let it be completed by normalisation and adding transition $\langle q', p'\rangle \xrightarrow{\mathfrak{K}} \langle q, p'\rangle$. Let $\mathcal{B}$ be the resulting automaton. It is enough to show that $\mathscr{L}(\mathcal{B}, \langle q', p'\rangle) \subseteq (R_{\text{in}}/E)^*\left([q]_E^{\mathcal{A}^{\not\mathfrak{K}}}\right)$. Remember that we use a modified version of normalisation that never reuses existing states, so $\mathscr{L}(\mathcal{B}, \langle q', p'\rangle) = \mathscr{L}(\mathcal{A}, r\sigma)$. We conclude thanks to Lemma 63. □

Theorem 64 and Lemma 66 can be summed as follows.

Theorem 67.
Innermost equational completion preserves R_{in}/E-coherence. 67 ■

This shows that under the assumption that $\mathcal{A}_0$ separates the classes of E, innermost completion will never add to the computed approximation a term that is not a descendant of $\mathscr{L}(\mathcal{A}_0)$ through R modulo E rewriting.

Theorem 68 (Precision).
Let E be a set of equations. Let $\mathcal{A}_0\mathcal{A}_\circ \times \mathcal{IRR}(R)$, where $\mathcal{A}_\circ$ has designated final states. We prune $\mathcal{A}_0$ of its non-accessible states. Suppose $\mathcal{A}_0$ separates the classes of E. Let R be any left-linear TRS. Let $\mathcal{A}$ be obtained from $\mathcal{A}_0$ after some steps of innermost equational completion. Then

$$\mathscr{L}(\mathcal{A}) \subseteq (R_{\text{in}}/E)^*(\mathscr{L}(\mathcal{A}_\circ \times \mathcal{IRR}(R))).$$ 68 ■

Proof.
Since $\mathcal{A}_0$ does not contain any $\mathfrak{K}$-marked transition, $\mathcal{A}_0^{\not\mathfrak{K}} = \mathcal{A}_0$. Thus, the third condition of (R_{in}/E)-coherence is trivial for $\mathcal{A}_0$ assuming the two first are met (for any TRS R). Plus, $\mathcal{A}_0$ separates the classes of E and is accessible, so it totally separates the classes of E, so $\mathcal{A}_0$ is indeed (R_{in}/E)-coherent. Theorem 67 show that this property is passed to $\mathcal{A}$, which yields the result. □

8. Related work

All tree automata completion-like techniques [Gen98, TKS00, Tak04, FGVTT04, BCHK09, GR10, Lis12] do not take evaluation strategies into account. They compute over-approximations of *all* reachable terms. Our contribution here is to propose the first algorithm computing over-approximations restricted to innermost reachable terms. Furthermore, the accuracy of the approximation w.r.t. innermost rewriting has been shown on a practical and theoretical point of view.

Dealing with reachable terms and strategies was first addressed in [RV02] in the exact case for innermost and outermost strategies but only for some restricted classes of TRSs, and also in [GGJ08]. As far as we know, the technique we propose is the first to over-approximate terms reachable by innermost rewriting for *any* left-linear TRSs. For instance, the Examples 17 and of Section 6 are in the scope of innermost completion but are outside of the classes of [RV02, GGJ08]. It is due to the fact that a right-hand side of a rule has two nested defined symbols and is not shallow.

Data flow analysis of higher-order functional programs is a long standing research topic [KI13, OR11, KO11, Jon87]. Used techniques ranges from tree grammars to specific formalisms: HORS, PMRS of ILTGs and can deal with higher-order functions. Higher-order functions are not in the scope of the work presented here, though it is possible with tree automata completion in general [GS13]. None of [KI13, OR11, KO11, Jon87], takes evaluation strategies into account and analysis results are thus coarse when program execution rely on a specific strategy.

9. Conclusion

In this paper, we have proposed a sound and precise algorithm over-approximating the set of terms reachable by innermost rewriting. As far as we know this is the first algorithm solving this problem for any left linear TRS and any regular initial set of terms. It is based on tree automata completion and equational abstractions with a set E of approximation equations. The algorithm also minimises the set of added transitions by completing the product automaton (between $\mathcal{A}_\circ$ and $\mathcal{I}_{\mathcal{RR}}(R)$). We proposed TimbukSTRAT [Tim14], a prototype implementation of this method.

The precision of the approximations have been shown on a theoretical and a practical point of view. On a theoretical point of view, we have shown that the approximation automaton recognises no more terms than those effectively reachable by innermost rewriting modulo the approximation E. On the practical side, unlike other techniques used to statically analyse functional programs [KI13, OR11, KO11, Jon87], completion can take the call-by-value strategy into account. As a result, for programs whose semantics highly depend on the evaluation strategy, innermost completion yields more accurate approximations. This should open new ways to statically analyse functional programs by taking evaluation strategies into account.

Approximations of sets of ancestors or descendants can improve existing termination techniques [GHWZ05, Mid02]. In the dependency pairs setting, such approximations can remove edges in a dependency graph by showing that there is no rewrite derivation from a pair to another. Besides, it has been shown that dependency pairs can prove innermost termination [GTSKF06]. In this case, *innermost* completion can more strongly prune the dependency graph: it can show that there is no *innermost* derivation from a pair to another. For instance, on the TRS:

$$\begin{array}{lll} choice(x,y) \to x & choice(x,y) \to y & eq(s(x),s(y)) \to eq(x,y) \\ eq(0,0) \to tt & eq(s(x),0) \to ff & eq(0,s(y)) \to ff \\ g(0,x) \to eq(x,x) & g(s(x),y) \to g(x,y) & f(ff,x,y) \to f(g(x,choice(x,y)),x,y) \end{array}$$

We can prove that any term of the form $f(g(t_1, choice(t_2,t_3)), t_4, t_5)$ cannot be rewritten (innermost) to a term of the form $f(ff, t_6, t_7)$ (for all terms $t_i \in T(\Sigma)$, $i = 1 \dots 7$). This proves that, in the dependency graph, there is no cycle on this pair. This makes the termination proof of this TRS simpler than what AProVE [APr] does: it needs more complex techniques, including proofs by induction. Simplification of termination proofs using innermost completion should

be investigated more deeply.

For further work, we want to improve and expand our implementation of innermost completion in order to design a strategy-aware and higher-order-able static analyser for OCaml. We also want to study if the innermost completion covers the decidable classes of [RV02], like standard completion does for many decidable classes [FGVTT04]. Another objective is to define a completion for the outermost strategy and thus deal with the call-by-need evaluation strategy, used in Haskell[8].

Acknowledgements The authors thank René Thiemann for providing the example of innermost terminating TRS submitted to AProVE, and Prof. Luke Ong, Jonathan Kochems and Robin Neatherway for their remarks.

References

[ABB+05] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santos Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the automated validation of internet security protocols and applications. In *CAV'2005*, volume 3576 of *LNCS*, pages 281–285. Springer, 2005.

[APr] AProVE. Automated Program Verification Environment. `http://aprove.informatik.rwth-aachen.de/`.

[BCHK09] Y. Boichut, R. Courbis, P.-C. Héam, and O. Kouchnarenko. Handling non left-linear rules when completing tree automata. *IJFCS*, 20(5), 2009.

[BGJL07] Y. Boichut, T. Genet, T. Jensen, and L. Leroux. Rewriting Approximations for Fast Prototyping of Static Analyzers. In *RTA*, volume 4533 of *LNCS*, pages 48–62. Springer, 2007.

[CR87] H. Comon and Jean-Luc Rémy. How to characterize the language of ground normal forms. Technical Report 676, INRIA-Lorraine, 1987.

[DF92] Olivier Danvy and Andrzej Filinski. Representing control: A study of the CPS transformation. *Mathematical Structures in Computer Science*, 2(4):361–391, 1992.

[FGVTT04] G. Feuillade, T. Genet, and V. Viet Triem Tong. Reachability Analysis over Term Rewriting Systems. *Journal of Automated Reasonning*, 33 (3-4):341–383, 2004.

[Gen98] T. Genet. Decidable approximations of sets of descendants and sets of normal forms. In *Proc. 9th RTA Conf., Tsukuba (Japan)*, volume 1379 of *LNCS*, pages 151–165. Springer-Verlag, 1998.

[GGJ08] Adria Gascon, Guillem Godoy, and Florent Jacquemard. Closure of Tree Automata Languages under Innermost Rewriting. In *8th International Workshop on Reduction*

[8] Note that in tree automata completion every computation is done only once, associated to a state and shared in the tree automaton, as in call-by-need.

Strategies in Rewriting and Programming (WRS), volume 237 of *ENTCS*, pages 23–38, Hagenberg, Autriche, 2008. Elsevier.

[GHWZ05] A. Geser, D. Hofbauer, J. Waldmann, and H. Zantema. On tree automata that certify termination of left-linear term rewriting systems. In *RTA'05*, volume 3467 of *LNCS*, pages 353–367. Springer, 2005.

[GK00] T. Genet and F. Klay. Rewriting for Cryptographic Protocol Verification. In *Proc. 17th CADE Conf., Pittsburgh (Pen., USA)*, volume 1831 of *LNAI*. Springer-Verlag, 2000.

[GR10] T. Genet and R. Rusu. Equational tree automata completion. *Journal of Symbolic Computation*, 45:574–597, 2010.

[GS13] T. Genet and Y. Salmon. Tree Automata Completion for Static Analysis of Functional Programs. Technical report, INRIA, 2013. `http://hal.archives-ouvertes.fr/hal-00780124/PDF/main.pdf`.

[GTSKF06] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and improving dependency pairs. *JAR*, 37(3):155–203, 2006.

[Jon87] N. D. Jones. Flow analysis of lazy higher-order functional programs. In S. Abramsky and C. Hankin, editors, *Abstract Interpretation of Declarative Languages*, pages 103–122. Ellis Horwood, Chichester, England, 1987.

[KI13] N. Kobayashi and A. Igarashi. Model-Checking Higher-Order Programs with Recursive Types. In *ESOP*, volume 7792 of *LNCS*, pages 431–450. Springer, 2013.

[KO11] J. Kochems and L. Ong. Improved Functional Flow and Reachability Analyses Using Indexed Linear Tree Grammars. In *RTA'11*, volume 10 of *LIPIcs*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2011.

[Lis12] A. Lisitsa. Finite Models vs Tree Automata in Safety Verification. In *RTA'12*, volume 15 of *LIPIcs*, pages 225–239, 2012.

[Mid02] A. Middeldorp. Approximations for strategies and termination. *ENTCS*, 70(6):1–20, 2002.

[OR11] L. Ong and S. Ramsay. Verifying higher-order functional programs with pattern-matching algebraic data types. In *POPL'11*, 2011.

[Plo75] Gordon D. Plotkin. Call-by-name, call-by-value and the lambda-calculus. *Theor. Comput. Sci.*, 1(2):125–159, 1975.

[RV02] P. Réty and J. Vuotto. Regular Sets of Descendants by some Rewrite Strategies. In *Proc. 13th RTA Conf., Copenhagen (Denmark)*, volume 2378 of *LNCS*. Springer-Verlag, 2002.

[Tak04] T. Takai. A Verification Technique Using Term Rewriting Systems and Abstract Interpretation. In *Proc. 15th RTA Conf., Aachen (Germany)*, volume 3091 of *LNCS*, pages 119–133. Springer, 2004.

[Tim14] Timbuk – reachability analysis and Tree Automata Calculations. IRISA / Université de Rennes 1, 2012–2014. http://www.irisa.fr/celtique/genet/timbuk/.

[TKS00] T. Takai, Y. Kaji, and H. Seki. Right-linear finite-path overlapping term rewriting systems effectively preserve recognizability. In *Proc. 11th RTA Conf., Norwich (UK)*, volume 1833 of *LNCS*. Springer-Verlag, 2000.

A. Normalisation

Here, we describe an algorithmic way to browse a configuration in order to normalise it, that is to create all the necessary intermediate transitions and the required new states. The discourse is quite fastidious and not unique, but this is not really important. Indeed, we either reuse existing transitions, and thus do not change the behaviour of the automaton, or create new states whose sole purpose is to recognise some subconfiguration of c. By definitions, these new states only appear in the new transitions that we add, they do not influence the previous behaviour of the automaton. The order in which they are added is therefore not important.

Definition 69 ($Norm_{\mathcal{A}}(c, \langle q, p\rangle)$).
Let $\mathcal{A}$ be an automaton, c be a non-trivial configuration and $\langle q, p\rangle$ be any state. Let $\Xi = (\xi_1, \ldots, \xi_K)$ be a postfix traversal of c (ξ_K is the root position). With Δ the set of transitions of $\mathcal{A}$, let us use an auxiliary function:

$$Norm_{\mathcal{A}}(c, \langle q, p\rangle) = NormAux_{\Delta}^{\Xi,1}(c, \langle q, p\rangle). \tag{1}$$

Now let us define $NormAux_{\cdot}^{\Xi,i}$. For i ranging from 1 to $K-1$, for any set of transitions Δ and any configuration d such that $d_{|\xi_i}$ is an elementary configuration, let

$$NormAux_{\Delta}^{\Xi,i}(d, \langle q, p\rangle) = \{d_{|\xi_i} \rightarrowtail \langle q', p'\rangle\} \cup NormAux_{\Delta \cup \{d_{|\xi_i} \rightarrowtail \langle q', p'\rangle\}}^{\Xi, i+1}(d[\langle q', p'\rangle]_{\xi_i}, \langle q, p\rangle) \tag{2}$$

where $\langle q', p'\rangle$ is such that $d_{|\xi_i} \rightarrowtail \langle q', p'\rangle$ in $\Delta/_{\mathrm{E}}$ (or, if there is no such state, q' is a new state, distinct from q, and p' is the state such that $\Pi_2\left(d_{|\xi_i}\right) \underset{\mathcal{IRR}(R)}{\overset{*}{\rightarrowtail}} p'$). Also, for any set of transitions Δ and any elementary configuration d, let

$$NormAux_{\Delta}^{\Xi,K}(d, \langle q, p\rangle) = \{d \rightarrowtail \langle q, p\rangle\}. \tag{3}$$

69 ◀

Remark. It is necessary to consider equivalence by E when searching for an already existing transition. Suppose we work with some $f(q_1)$ and have $q_1 \underset{\Delta}{\overset{\mathrm{E}}{\rightleftarrows}} q_2$ and $f(q_2) \underset{\Delta}{\rightarrowtail} q'$: we do not want to create a new state here, but reuse q' and set $f(q_1) \rightarrowtail q'$.

B. Completed automaton for the `sum` function

Let R be the TRS:

(1) $0 + X \to X$
(2) $s(X) + Y \to s(X + Y)$
(3) $sumList(X, Y) \to cons(X + Y, sumList(X + Y, s(Y)))$
(4) $nth(0, cons(X, Y)) \to X$
(5) $nth(s(X), cons(Y, Z)) \to nth(X, Z)$
(6) $sum(X) \to nth(X, sumList(0, 0))$

and $E = \{cons(x, cons(y, z)) = cons(y, z), (x+y)+z = x+y\}$. Starting from the product automaton having transitions: $0 \rightarrowtail \langle q_0, p_i\rangle$, $s(\langle q_0, p_i\rangle) \rightarrowtail \langle q_0, p_i\rangle$, $sum(\langle q_0, p_i\rangle) \rightarrowtail \langle q_1, p_{\text{red}}\rangle$. We successively solve the following critical pairs.

B.1. First completion step

The first completion step solves only one critical pair which is:

$$\begin{array}{ccc} sum(\langle q_0, p_i\rangle) & \xrightarrow{R} & nth(\langle q_0, p_i\rangle, sumList(0,0)) \\ {\scriptstyle \mathcal{A}^0}\downarrow{\scriptstyle *} & & {\scriptstyle *}\vdots{\scriptstyle \mathcal{A}^1} \\ \langle q_1, p_{\text{red}}\rangle & \underset{\mathcal{A}^1}{\overset{\aleph}{\dashleftarrow}} & \langle q_3, p_{\text{red}}\rangle \end{array}$$

The set of transitions added to $\mathcal{A}_0$ to obtain $\mathcal{A}_1$ consists of $nth(\langle q_0, p_i\rangle, \langle q_2, p_{\text{red}}\rangle) \rightarrowtail \langle q_3, p_{\text{red}}\rangle$, $sumList(\langle q_0, p_i\rangle, \langle q_0, p_i\rangle) \rightarrowtail \langle q_2, p_{\text{red}}\rangle$, and $\langle q_3, p_{\text{red}}\rangle \overset{\aleph}{\rightarrowtail} \langle q_1, p_{\text{red}}\rangle$.

B.2. Second completion step

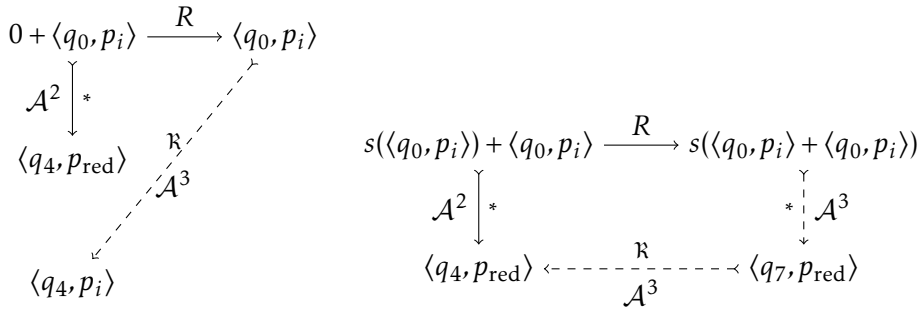
The second completion step solves one critical pair:

$$\begin{array}{ccc} sumList(\langle q_0, p_i\rangle, \langle q_0, p_i\rangle) & \xrightarrow{R} & cons(\langle q_0, p_i\rangle + \langle q_0, p_i\rangle, sumList(\langle q_0, p_i\rangle + \langle q_0, p_i\rangle, s(\langle q_0, p_i\rangle))) \\ {\scriptstyle \mathcal{A}^1}\downarrow{\scriptstyle *} & & {\scriptstyle *}\vdots{\scriptstyle \mathcal{A}^2} \\ \langle q_2, p_{\text{red}}\rangle & \underset{\mathcal{A}^2}{\overset{\aleph}{\dashleftarrow}} & \langle q_6, p_{\text{red}}\rangle \end{array}$$

Normalisation of the corresponding transition results in the following set of new transitions to be added to $\mathcal{A}_1$: $cons(\langle q_4, p_{\text{red}}\rangle, \langle q_5, p_{\text{red}}\rangle) \rightarrowtail \langle q_6, p_{\text{red}}\rangle$, $\langle q_0, p_i\rangle + \langle q_0, p_i\rangle \rightarrowtail \langle q_4, p_{\text{red}}\rangle$, $sumList(\langle q_4, p_{\text{red}}\rangle, \langle q_0, p_i\rangle) \rightarrowtail \langle q_5, p_{\text{red}}\rangle$ and $\langle q_6, p_{\text{red}}\rangle \overset{\aleph}{\rightarrowtail} \langle q_2, p_{\text{red}}\rangle$.

B.3. Third completion step

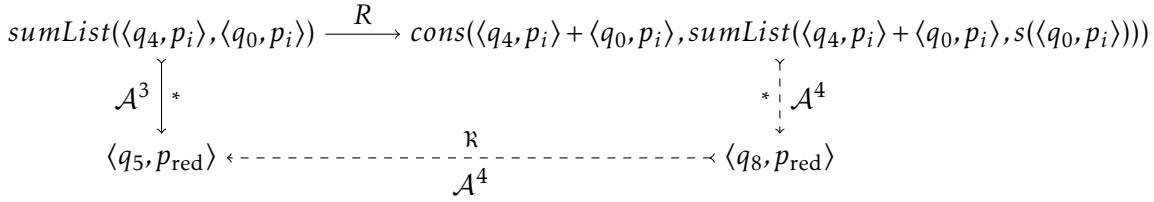
The third completion step solves two critical pairs. The first critical pairs are:



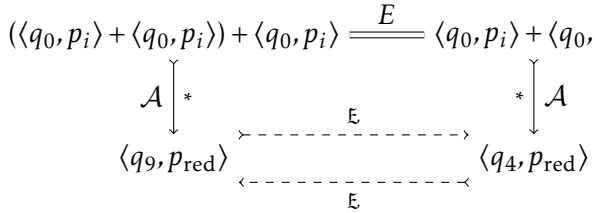
For solving the second critical pair, since the transition $\langle q_0, p_i\rangle + \langle q_0, p_i\rangle \rightarrowtail \langle q_4, p_{\text{red}}\rangle$ already belong to $\mathcal{A}_2$, it is enough to add the transitions: $s(\langle q_4, p_{\text{red}}\rangle) \rightarrowtail \langle q_7, p_{\text{red}}\rangle$ and $\langle q_7, p_{\text{red}}\rangle \overset{\mathfrak{K}}{\rightarrowtail} \langle q_4, p_{\text{red}}\rangle$. For solving the first critical pair, it is enough to add the transition $\langle q_0, p_i\rangle \overset{\mathfrak{K}}{\rightarrowtail} \langle q_4, p_i\rangle$ and to add supplementary transitions, as mentioned in Definition 47. Because of the separation between states $\langle q_4, p_{\text{red}}\rangle$ and $\langle q_4, p_i\rangle$, it is necessary to add a copy of all transitions with a state $\langle q_4, p_{\text{red}}\rangle$ in their left-hand side where we replace it by state $\langle q_4, p_i\rangle$ to . We thus add the transitions: $cons(\langle q_4, p_i\rangle, \langle q_5, p_{\text{red}}\rangle) \rightarrowtail \langle q_6, p_{\text{red}}\rangle$, $sumList(\langle q_4, p_i\rangle, \langle q_0, p_i\rangle) \rightarrowtail \langle q_5, p_{\text{red}}\rangle$ and $s(\langle q_4, p_i\rangle) \rightarrowtail \langle q_7, p_i\rangle$.

B.4. Fourth critical pair

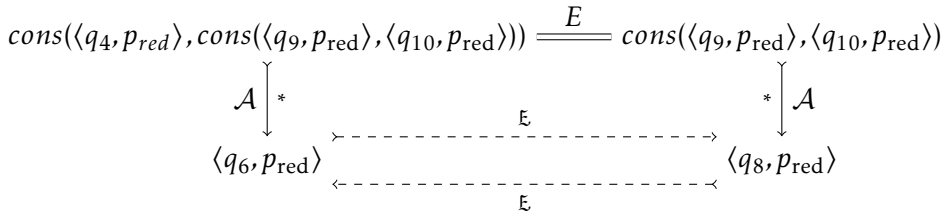
The fourth completion step solves one critical pair and applies two equations. The critical pair is:



The transitions added to $\mathcal{A}_3$ are thus: $\langle q_4, p_i\rangle + \langle q_0, p_i\rangle \rightarrowtail \langle q_9, p_{\text{red}}\rangle$, $sumList(\langle q_9, p_{\text{red}}\rangle, \langle q_0, p_i\rangle) \rightarrowtail \langle q_{10}, p_{\text{red}}\rangle$, $cons(\langle q_9, p_{\text{red}}\rangle, \langle q_{10}, p_{\text{red}}\rangle) \rightarrowtail \langle q_8, p_{\text{red}}\rangle$ and $\langle q_8, p_{\text{red}}\rangle \overset{\mathfrak{K}}{\rightarrowtail} \langle q_5, p_{\text{red}}\rangle$. Now, with equation $(x + y) + z = x + y$ of E, we can find a first equational simplification position:



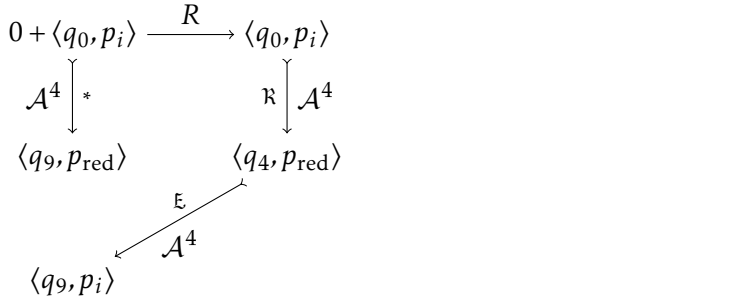
This results in the addition of transitions $\langle q_9, p\rangle \overset{\mathfrak{E}}{\rightleftarrows} \langle q_4, p\rangle$ for all state p. Similarly the equation $cons(x, cons(y, z)) = cons(y, z)$ of E finds an equational simplification position:

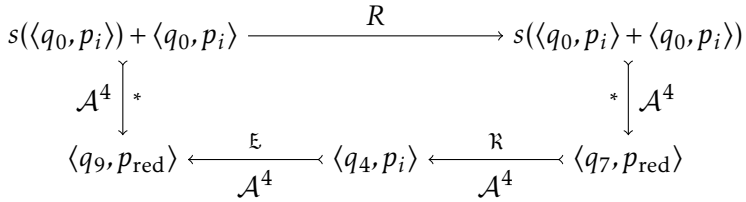


This results in the addition of transitions $\langle q_6, p\rangle \overset{\mathrm{E}}{\rightleftarrows} \langle q_8, p\rangle$ for all state p.

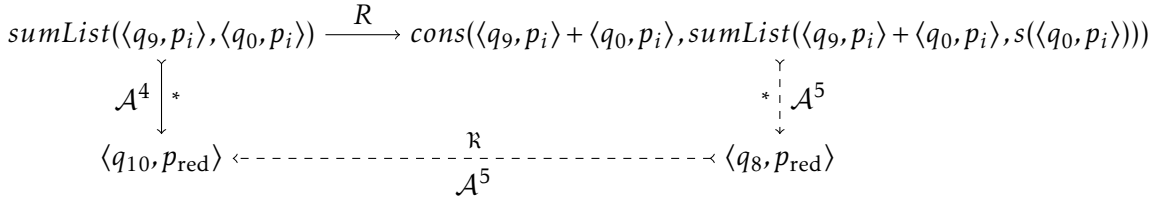
B.5. Fith completion step

Now all critical pairs on rules (1) and (2) are joinable:

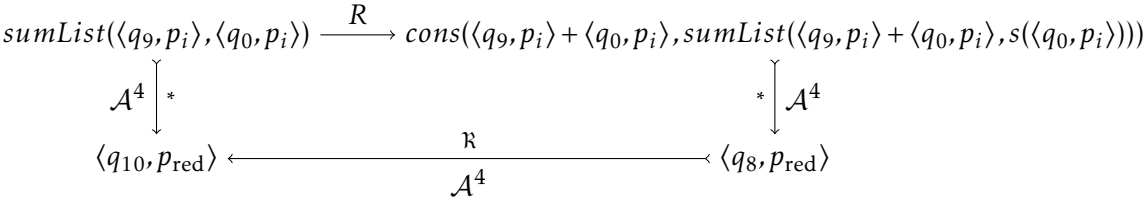




There remains a last critical pair which is:



However, using transitions of $\mathcal{A}_4$ the right-hand side of the critical pair can be recognised into state $\langle q_6, p_{\mathrm{red}}\rangle$. More in details, the subterm $\langle q_9, p_i\rangle + \langle q_0, p_i\rangle$ can be normalised into $\langle q_4, p_{\mathrm{red}}\rangle$ and the subterm $sumList(\langle q_9, p_i\rangle, s(\langle q_0, p_i\rangle))$ can be normalised into $\langle q_5, p_{\mathrm{red}}\rangle$. Finally $cons(\langle q_4, p_i\rangle, \langle q_5, p_{\mathrm{red}}\rangle) \rightarrowtail \langle q_8, p_{\mathrm{red}}\rangle$ belongs to transitions of $\mathcal{A}_4$. Furthermore, $\langle q_6, p_{\mathrm{red}}\rangle \overset{\mathrm{E}}{\rightarrowtail} \langle q_8, p_{\mathrm{red}}\rangle$. Finally the only transition to add is thus $\langle q_8, p_{\mathrm{red}}\rangle \overset{\mathrm{R}}{\rightarrowtail} \langle q_{10}, p_{\mathrm{red}}\rangle$. Since no new transition is added on symbol $sumList$, this ends the completion. All remaining critical pairs are joinable, in particular:



The transitions of the completed automaton are:
$0 \rightarrowtail \langle q_0, p_i\rangle$
$s(\langle q_0, p_i\rangle) \rightarrowtail \langle q_0, p_i\rangle$
$sum(\langle q_0, p_i\rangle) \rightarrowtail \langle q_1, p_{\text{red}}\rangle$
$nth(\langle q_0, p_i\rangle, \langle q_2, p_{\text{red}}\rangle) \rightarrowtail \langle q_3, p_{\text{red}}\rangle$
$sumList(\langle q_0, p_i\rangle, \langle q_0, p_i\rangle) \rightarrowtail \langle q_2, p_{\text{red}}\rangle$
$\langle q_3, p_{\text{red}}\rangle \overset{\mathfrak{R}}{\rightarrowtail} \langle q_1, p_{\text{red}}\rangle$
$cons(\langle q_4, p_{\text{red}}\rangle, \langle q_5, p_{\text{red}}\rangle) \rightarrowtail \langle q_6, p_{\text{red}}\rangle$
$\langle q_0, p_i\rangle + \langle q_0, p_i\rangle \rightarrowtail \langle q_4, p_{\text{red}}\rangle$
$sumList(\langle q_4, p_{\text{red}}\rangle, \langle q_0, p_i\rangle) \rightarrowtail \langle q_5, p_{\text{red}}\rangle$
$\langle q_6, p_{\text{red}}\rangle \overset{\mathfrak{R}}{\rightarrowtail} \langle q_2, p_{\text{red}}\rangle$
$s(\langle q_4, p_{\text{red}}\rangle) \rightarrowtail \langle q_7, p_{\text{red}}\rangle$
$\langle q_7, p_{\text{red}}\rangle \overset{\mathfrak{R}}{\rightarrowtail} \langle q_4, p_{\text{red}}\rangle$
$\langle q_0, p_i\rangle \overset{\mathfrak{R}}{\rightarrowtail} \langle q_4, p_i\rangle$
$cons(\langle q_4, p_i\rangle, \langle q_5, p_{\text{red}}\rangle) \rightarrowtail \langle q_6, p_{\text{red}}\rangle$
$sumList(\langle q_4, p_i\rangle, \langle q_0, p_i\rangle) \rightarrowtail \langle q_5, p_{\text{red}}\rangle$
$s(\langle q_4, p_i\rangle) \rightarrowtail \langle q_7, p_i\rangle$
$\langle q_4, p_i\rangle + \langle q_0, p_i\rangle \rightarrowtail \langle q_9, p_{\text{red}}\rangle$
$sumList(\langle q_9, p_{\text{red}}\rangle, \langle q_0, p_i\rangle) \rightarrowtail \langle q_{10}, p_{\text{red}}\rangle$
$sumList(\langle q_9, p_i\rangle, \langle q_0, p_i\rangle) \rightarrowtail \langle q_{10}, p_{\text{red}}\rangle$
$cons(\langle q_9, p_{\text{red}}\rangle, \langle q_{10}, p_{\text{red}}\rangle) \rightarrowtail \langle q_8, p_{\text{red}}\rangle$
$\langle q_8, p_{\text{red}}\rangle \overset{\mathfrak{R}}{\rightarrowtail} \langle q_5, p_{\text{red}}\rangle$
$\langle q_9, p\rangle \overset{\mathfrak{E}}{\rightleftarrows} \langle q_4, p\rangle$ for all p
$\langle q_6, p\rangle \overset{\mathfrak{E}}{\rightleftarrows} \langle q_8, p\rangle$ for all p
$\langle q_8, p_{\text{red}}\rangle \overset{\mathfrak{R}}{\rightarrowtail} \langle q_{10}, p_{\text{red}}\rangle$

Note that this completed automaton recognises, in particular, all infinite list of natural number, *i.e.* terms of the form $cons(a, cons(b, cons(c, \ldots sumList(d, e))))$ where a, b, c, d, e are any natural numbers. For instance, the term $cons(0, cons(0, cons(0, cons(0, sumList(0, 0)))))$ is recognised in the following way:

$cons(0, cons(0, cons(0, cons(0, sumList(0, 0)))))$
$\rightarrowtail^* cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, sumList(\langle q_0, p_i\rangle, \langle q_0, p_i\rangle)))))$
$\overset{\mathfrak{R}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, sumList(\langle q_4, p_i\rangle, \langle q_0, p_i\rangle)))))$
$\overset{\mathfrak{R}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, sumList(\langle q_4, p_i\rangle, \langle q_0, p_i\rangle)))))$

$$
\begin{aligned}
&\overset{\mathrm{E}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, sumList(\langle q_9, p_i\rangle, \langle q_0, p_i\rangle)))))\\
&\rightarrowtail cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, \langle q_{10}, p_{\mathrm{red}}\rangle))))\\
&\rightarrowtail cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, \langle q_{10}, p_{\mathrm{red}}\rangle))))\\
&\overset{\mathrm{R}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_4, p_i\rangle, \langle q_{10}, p_{\mathrm{red}}\rangle))))\\
&\overset{\mathrm{E}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_9, p_i\rangle, \langle q_{10}, p_{\mathrm{red}}\rangle))))\\
&\rightarrowtail cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, \langle q_8, p_{\mathrm{red}}\rangle)))\\
&\overset{\mathrm{R}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, \langle q_5, p_{\mathrm{red}}\rangle)))\\
&\overset{\mathrm{R}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, cons(\langle q_4, p_i\rangle, \langle q_5, p_{\mathrm{red}}\rangle)))\\
&\overset{\mathrm{R}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, \langle q_6, p_{\mathrm{red}}\rangle))\\
&\overset{\mathrm{E}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, \langle q_8, p_{\mathrm{red}}\rangle))\\
&\overset{\mathrm{R}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, \langle q_5, p_{\mathrm{red}}\rangle))\\
&\overset{\mathrm{R}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, cons(\langle q_4, p_i\rangle, \langle q_5, p_{\mathrm{red}}\rangle))\\
&\overset{\mathrm{R}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, \langle q_6, p_{\mathrm{red}}\rangle)\\
&\overset{\mathrm{E}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, \langle q_8, p_{\mathrm{red}}\rangle)\\
&\overset{\mathrm{R}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, \langle q_5, p_{\mathrm{red}}\rangle)\\
&\overset{\mathrm{R}}{\rightarrowtail} cons(\langle q_4, p_i\rangle, \langle q_5, p_{\mathrm{red}}\rangle)\\
&\rightarrowtail \langle q_6, p_{\mathrm{red}}\rangle\\
&\overset{\mathrm{R}}{\rightarrowtail} \langle q_2, p_{\mathrm{red}}\rangle
\end{aligned}
$$