HAL
open science

Reachability Analysis of Innermost Rewriting

Thomas Genet, Yann Salmon

To cite this version:

Thomas Genet, Yann Salmon. Reachability Analysis of Innermost Rewriting. [Research Report] 2013. hal-00848260v1

HAL Id: hal-00848260

https://inria.hal.science/hal-00848260v1

Submitted on 25 Jul 2013 (v1), last revised 10 Feb 2014 (v5)

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

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

Reachability Analysis of Innermost Rewriting

Thomas Genet* Yann Salmon†

July 25, 2013

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. For such applications, the precision of the approximation is crucial. However, such approximations are coarse when the rewriting system is used under a specific strategy. This is due to the fact that techniques approximating reachable terms do not take rewriting strategies into account. Thus, they produce approximations containing terms that are not reachable w.r.t. the rewriting 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. We also show that it noticeably improves the accuracy of static analysis for functional programming languages using the call-by-value evaluation strategy.

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 only one to have tackled this goal. He gives some sufficient conditions on L_0 and R for $R^*_{strat}(L_0)$ to be recognised by a tree automaton $\mathcal{A}^*$, where $strat$ can be the innermost or the outermost strategy. However, those restrictions

*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

on R and L_0 are strong and generally incompatible with programs seen as TRS. Moreover, the proposed technique is not abble 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 and precision theorems. Section 6 demonstrates how our new technique can effectively give more precise results on a functional program.

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 ◀

Definition 4 (Context).

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

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

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. $\mathrm{Pos}(f(t_1,\ldots,t_k)) = \{\Lambda\} \cup \bigcup_{i=1}^{k} \{i\}.\mathrm{Pos}(t_i)$. 5 ◀

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

1. $\mathrm{Pos}_\square(\square) = \Lambda$
2. $\mathrm{Pos}_\square(f(C_1,\ldots,C_k)) = i.\mathrm{Pos}_\square(C_i)$, 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 $\mathrm{Pos}_\square(C) = p$ 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 ◀

Definition 17 (Determinism, Completeness).

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. 17 ◀

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 18.
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 $\mathfrak{K}$. 18 ◀

Definition 19.
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 $q' \underset{\mathcal{A}}{\overset{\mathfrak{E}}{\rightarrowtail}} q$. 19 ◀

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 20.
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

$$\begin{aligned}&\left\{f([q_1]_{\mathfrak{E}}, \ldots, [q_k]_{\mathfrak{E}}) \rightarrowtail [q']_{\mathfrak{E}} \;\middle|\; f(q_1, \ldots, q_k) \rightarrowtail q' \in \Delta\right\}\\ \cup &\left\{[q]_{\mathfrak{E}} \rightarrowtail [q']_{\mathfrak{E}} \;\middle|\; q \rightarrowtail q' \in \Delta \wedge [q]_{\mathfrak{E}} \neq [q']_{\mathfrak{E}}\right\}.\end{aligned}$$ 20 ◀

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 21 (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 $Q = Q_1 \times Q_2$. 21 ◀

Definition 22 (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

$$\begin{aligned}\Delta = &\left\{f(\langle q_1, p_1\rangle, \ldots, \langle q_k, p_k\rangle) \rightarrowtail \langle q', p'\rangle \;\middle|\; f(q_1, \ldots, q_k) \rightarrowtail q' \in \Delta_{\mathcal{A}} \wedge f(p_1, \ldots, p_k) \rightarrowtail p' \in \Delta_{\mathcal{B}}\right\}\\ &\cup \left\{\langle q, p\rangle \rightarrowtail \langle q', p'\rangle \;\middle|\; q \rightarrowtail q' \in \Delta_{\mathcal{A}}\right\} \cup \left\{\langle q, p\rangle \rightarrowtail \langle q, p'\rangle \;\middle|\; p \rightarrowtail p' \in \Delta_{\mathcal{B}}\right\}.\end{aligned}$$ 22 ◀

Definition 23 (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. 23 ◀

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 $\to_R$. 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 24 (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 $s \to_{R_{\text{in}}} t$, if $s \to_R t$ and each strict subterm of the redex in s is a R-normal form. We define $\to^*_{R_{\text{in}}}$, $R_{\text{in}}(L)$ and $R^*_{\text{in}}(L)$ as in Definitions 8 and 9. 24 ◀

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 25 ([CR87]).
Let R be a left-linear TRS. There is a deterministic and complete tree automaton $\mathcal{IRR}(R)$ such that $\mathscr{L}(\mathcal{IRR}(R)) = \textsc{Irr}(R)$ and with a distinguished state p_{red} such that $\mathscr{L}(\mathcal{IRR}(R), p_{\text{red}}) = T(\Sigma) \setminus \textsc{Irr}(R)$. 25 ■

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

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 26 (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. $r\sigma \overset{*}{\underset{\mathcal{A}}{\not\rightarrowtail}} q$. 26 ◀

We will want 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 normalise this configuration by adding intermediate states. For example, to set $f(g(a,b)) \rightarrowtail q'$, we will first add $a \rightarrowtail q_a$, then, $b \rightarrowtail q_b$, $g(q_a, q_b) \rightarrowtail q_g$ and finally $f(q_g) \rightarrowtail q'$, exploring $f(g(a,b))$ with a postfix traversal. At each of these steps, we will look whether we could reuse an already existing transition; if not, we will add a new state.

Definition 27 ($Norm_{\mathcal{A}}(c,q)$).

Let $\mathcal{A}$ be an automaton, c be a non-trivial configuration and q 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:

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

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

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

where q' is such that $d_{|\xi_i} \rightarrowtail q'$ in $\Delta/\mathfrak{E}$ (or, if there is no such state, q' is a new state, distinct from q). Also, for any set of transitions Δ and any elementary configuration d, let

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

27 ◀

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

Definition 28 (Completion of a critical pair).

A critical pair $CP = (\ell \to r, \sigma, q)$ in automaton $\mathcal{A}$ is completed by first computing $N = Norm_{\mathcal{A}/\mathfrak{R}}(r\sigma, q')$ 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 $q' \overset{\mathfrak{R}}{\rightarrowtail} q$. If $r\sigma$ is a trivial configuration (ie. r is just a variable), only transition $r\sigma \overset{\mathfrak{R}}{\rightarrowtail} q$ is added. 28 ◀

(a) A critical pair

(b) Situation of application

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

Definition 29 (Step of completion).

The automaton resulting from completion of a list comprising of one critical pair CP is noted $\mathcal{J}_R^{(CP)}(\mathcal{A})$. For a list of zero critical pair, we set $\mathcal{J}_R^{()}(\mathcal{A}) = \mathcal{A}$ and for any list LCP of critical pairs (of $\mathcal{A}$) whose head is CP and tail is LCP', $\mathcal{J}_R^{LCP}(\mathcal{A}) = \mathcal{J}_R^{LCP'}\left(\mathcal{J}_R^{(CP)}(\mathcal{A})\right)$. A step of completion of automaton $\mathcal{A}$ consists in producing automaton $\mathcal{C}_R(\mathcal{A}) = \mathcal{J}_R^{\mathcal{CP}}(\mathcal{A})$, where $\mathcal{CP}$ is a list of all critical pairs of $\mathcal{A}$. 29 ◀

Example 30.

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 $R = \{f(n) \to n, f(c(s(X), Y)) \to c(s(X), f(Y)), f(c(a(X), Y)) \to c(a(X), f(Y)), f(c(0, Y)) \to f(Y), a(s(X)) \to X, s(a(X)) \to X\}$. 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 $R(\mathscr{L}(\mathcal{A}_0, q_f)) = \{f(c(0, n)), c(a(s(0)), f(n))\}$.

There is a critical pair CP_1 in $\mathcal{A}_0$ with the rule $f(c(a(X), Y)) \to c(a(X), f(Y))$, 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 $q_{N2} \overset{\mathfrak{R}}{\rightarrowtail} q_f$, and have produced $\mathcal{J}_R^{(PC_1)}(\mathcal{A}_0)$.

Another critical pair is CP_2 in $\mathcal{A}_0$ with the rule $a(s(X)) \to X$, 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 $q_0 \overset{\mathfrak{R}}{\rightarrowtail} q_a$, 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 $f(n) \to n$, the empty substitution and state q_{N1}. 30 ◁

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 31 (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. $q_1 \overset{E}{\underset{\mathcal{A}}{\not\rightarrowtail}} q_2$. 31 ◀

Definition 32 (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 $q_1 \overset{\mathfrak{E}}{\rightarrowtail} q_2$ and $q_2 \overset{\mathfrak{E}}{\rightarrowtail} q_1$ to $\mathcal{A}$. This produces a new automaton $\mathcal{A}'$ and we note $\mathcal{A} \rightsquigarrow_E \mathcal{A}'$. 32 ◀

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 $\mathcal{A}/_{\mathfrak{E}}$ (see definition 20) and finally choosing a representative (here q_1 for the class $\{q_1, q_2\}$) of each equivalence class of states.

Definition 33 (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 $\mathcal{A}$. 33 ◀

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

Definition 34 (Step of equational completion).
A step of equational completion is the composition of a step of exact completion, then equational simplification: $\mathcal{CE}_{R,E}(\mathcal{A}) = \mathcal{S}_E(\mathcal{C}_R(\mathcal{A}))$. 34 ◀

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

Definition 35 (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 $\mathscr{L}(\mathcal{A}_0, q) \subseteq [s]_E$. 35 ◀

3.3. Known results

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

Theorem 36 (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

$$\mathscr{L}(\mathcal{A}_*) \supseteq R^*(\mathscr{L}(\mathcal{A}_0)).$$ 36 ■

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

Lemma 37.
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 $\mathcal{A}^{\not\mathfrak{E}}$ is E-coherent and moreover, for all state q of $\mathcal{A}$, there exists $s \in \mathscr{L}(\mathcal{A}^{\not\mathfrak{E}}, q)$ such that $\mathscr{L}(\mathcal{A}, q) \subseteq (R/E)^*(s)$. 37 ■

Such an automaton is said to be R/E-coherent. The intuition behind this is the following: in the tree automaton, $\mathfrak{R}$-transitions represent rewriting steps and transitions of $\mathcal{A}^{\not\mathfrak{R}}$ 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 $\mathcal{A}^{\not\mathfrak{R}}$ then they belong to the same E-equivalence class. Otherwise, if at least one $\mathfrak{R}$-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 $\mathfrak{R}$-transition, $\mathcal{A}_0$ is R/E-coherent if and only if it is E-coherent.

Theorem 38 (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

$$\mathscr{L}(\mathcal{A}) \subseteq (R/E)^*(\mathscr{L}(\mathcal{A}_0))$$ 38 ■

4. Adaptation to innermost strategies

4.1. Introduction

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 39.
Let us look at Example 30. 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 30. 39 ◁

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 25). 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 40 (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 $\Pi_2(c) \overset{*}{\underset{\mathcal{IRR}(R)}{\rightarrowtail}} p$. 40 ◀

The problem is, then, to update this product after each completion step: each completion step produces new states and new transitions that are not in $\mathcal{A}$ and thus not covered by the product. A naive way to do this would be to re-compute the product after each new completion step. However, this is rather inefficient: if N_A is the number of transitions of $\mathcal{A}$ and N_I is the number

of transition of $\mathcal{IRR}(R)$, the product automaton can have $N_A \times N_I$ transitions, and this product will be done after each step of completion. Hence, for n steps of completion, the number of transitions can be exponential w.r.t. n, i.e. $N_A \times (N_I)^n$. We propose a more efficient solution where completion only produces *transitions of the product automaton*.

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. Equational simplification

Definition 41 (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. $s\theta \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} \langle q_1, p_1\rangle$,
2. $t\theta \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} \langle q_2, p_2\rangle$,
3. $\langle q_1, p_1\rangle \overset{\varepsilon}{\underset{\mathcal{A}}{\not\rightleftarrows}} \langle q_2, p_2\rangle$ and
4. $p_1 = p_2$. 41 ◀

Definition 42 (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 $\langle q_1, p_1\rangle \overset{\varepsilon}{\rightarrowtail} \langle q_2, p_1\rangle$ and $\langle q_2, p_1\rangle \overset{\varepsilon}{\rightarrowtail} \langle q_1, p_1\rangle$ to $\mathcal{A}$. 42 ◀

Lemma 43.

Applying an equation preserves consistency with $\mathcal{IRR}(R)$. 43 ■

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 $c \overset{*}{\underset{\mathcal{B}}{\rightarrowtail}} \langle q, p\rangle$ where c is a configuration and $\langle q, p\rangle$ is a state of $\mathcal{B}$. If there is no occurrence, then $c \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} \langle q, p\rangle$ and by consistency of $\mathcal{A}$, $\Pi_2(c) \overset{*}{\underset{\mathcal{IRR}(R)}{\rightarrowtail}} p$.

Suppose the property is true for some k and there is a context C on $T(\Sigma, Q)$ such that $c \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} C[\langle q_1, p_1\rangle] \underset{\mathcal{B}}{\rightarrowtail} C[\langle q_2, p_1\rangle] \overset{*}{\underset{\mathcal{B}}{\rightarrowtail}} \langle q, p\rangle$ 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 $c_1 \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} \langle q_1, p_1\rangle$, and therefore $\Pi_2(c_1) \overset{*}{\underset{\mathcal{IRR}(R)}{\rightarrowtail}} p_1$. Second, by induction hypothesis, $\Pi_2(C[p_1]) \overset{*}{\underset{\mathcal{IRR}(R)}{\rightarrowtail}} p$. Finally, $\Pi_2(c) \overset{*}{\underset{\mathcal{IRR}(R)}{\rightarrowtail}} \Pi_2(C[p_1]) \overset{*}{\underset{\mathcal{IRR}(R)}{\rightarrowtail}} p$. □

Remark. The condition 4 ($p_1 = p_2$) of definition 41 is obviously necessary for this lemma. We will explain why consistency is important after having defined the notion of innermost critical pair (Definition 44).

4.3. 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 $\mathcal{IRR}(R)$.

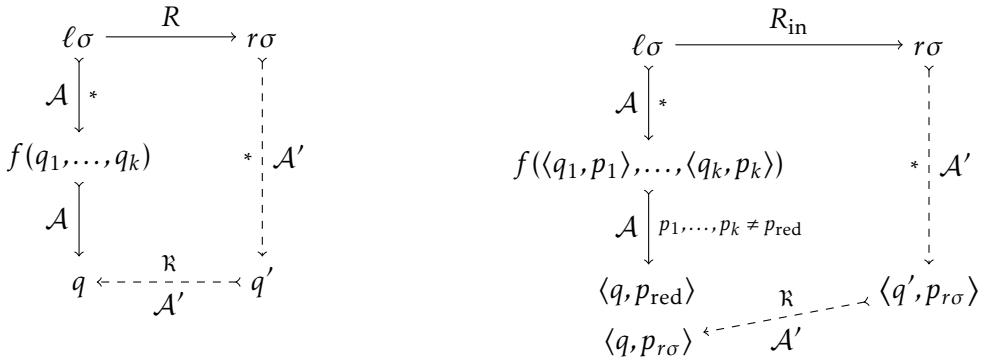
Remark. We can now explain why consistency is important and what would happen if we allowed equations to be applied without regard for condition 4 of Definition 41. Take $R = \{f(a) \to w, g(f(b)) \to c\}$, $E = \{a = b\}$ and $\mathcal{A}_\circ = \{a \rightarrowtail q_a, b \rightarrowtail q_b, f(q_a) \rightarrowtail q_{fa}, g(q_{fa}) \rightarrowtail q_{gfa}\}$. 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 41 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 $g(f(b)) \in \mathscr{L}(\mathcal{A}, \left\langle q_{gfa}, p_{\text{red}}\right\rangle)$ 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.

Definition 45 (Normalisation).
This is like in Definition 27, except we deal with pairs $\langle q, p\rangle$ instead of just q. In particular, when we do not find an existing $\langle q', p'\rangle$ such that $d_{|\xi_i} \underset{\Delta/\xi}{\rightarrowtail} \langle q', p'\rangle$, we take q' a new state, and p' the state such that $\Pi_2\left(d_{|\xi_i}\right) \overset{*}{\underset{\mathcal{IRR}(R)}{\rightarrowtail}} p'$. 45 ◀



(a) Classical

(b) Innermost

Figure 2: Comparison of classical and innermost critical pairs

Definition 46 (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 $N = Norm_{\mathcal{A}^{\mathfrak{K}}}(r\sigma, \langle q', p_{r\sigma}\rangle)$ 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 $\langle q', p_{r\sigma}\rangle \overset{\mathfrak{K}}{\rightarrowtail} \langle q, p_{r\sigma}\rangle$. If $r\sigma$ is a trivial configuration (ie. r is just a variable), only transition $r\sigma \overset{\mathfrak{K}}{\rightarrowtail} \langle q, \Pi_2(r\sigma)\rangle$ is added.

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

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

we add (if it is not present) a transition

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

with $f(\dots,p_{r\sigma},\dots) \underset{\mathcal{IRR}(R)}{\rightarrowtail} p'''$. These new transitions are in turn recursively considered for the supplementary operations. 46 ◀

Definition 47 (Step of exact completion).

This is the same definition as Definition 29, but using Definition 46 instead of Definition 28 for each critical pair. 47 ◀

Example 48.

Let us look again at Example 30. We have $R_{\text{in}}(\mathcal{L}(\mathcal{A}_0, q_f)) = \{f(c(0,n))\}$. Note that $\mathcal{IRR}(R)$ has states $p_0, p_a, p_s, p_n, p_c, p_?, p_{\text{red}}$ and transitions

$$\begin{aligned}
&\{n \rightarrowtail p_n, 0 \rightarrowtail p_0, s(p_a) \rightarrowtail p_{\text{red}}, a(p_s) \rightarrowtail p_{\text{red}}\}\\
\cup &\bigcup_{p\neq p_a} \{s(p) \rightarrowtail p_s\}\\
\cup &\bigcup_{p\neq p_s} \{a(p) \rightarrowtail p_a\}\\
\cup &\bigcup_{p} \{c(p_0,p) \rightarrowtail p_c, c(p_s,p) \rightarrowtail p_c, c(p_a,p) \rightarrowtail p_c, c(p_n,p) \rightarrowtail p_?, c(p_?,p) \rightarrowtail p_?\}
\end{aligned}$$

$$\cup \{f(p_n) \rightarrowtail p_{\text{red}}, f(p_c) \rightarrowtail p_{\text{red}}\}$$
$$\cup \bigcup_{p \neq p_n, p_c} \{f(p) \rightarrowtail p_?\}.$$

The critical pair corresponding to CP_1 of Example 30, with 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 $\left\langle q_f, p_{\text{red}} \right\rangle$, is not an innermost critical pair because the recognition path is

$$\begin{aligned} f(c(a(\langle q_s, p_s \rangle), \langle q_n, p_n \rangle)) &\rightarrowtail f(c(\langle q_a, p_{\text{red}} \rangle, \langle q_n, p_n \rangle)) \\ &\rightarrowtail f(\langle q_c, p_{\text{red}} \rangle) \\ &\rightarrowtail \left\langle q_f, p_{\text{red}} \right\rangle, \end{aligned}$$

and so there is a p_{red} at depth 1. 48 ◁

Lemma 49.
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 $\mathcal{IRR}(R)$. 49 ■

Proof.
$\mathcal{A}_\circ \times \mathcal{IRR}(R)$ is consistent with $\mathcal{IRR}(R)$ by construction. Lemma 43 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 $\Pi_2\left(d_{|\xi_i}\right) \overset{*}{\underset{\mathcal{IRR}(R)}{\rightarrowtail}} p'_i$. 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 $r\sigma \overset{*}{\underset{\mathcal{IRR}(R)}{\rightarrowtail}} p'$.

The same goes for the supplementary operations. □

Lemma 50.
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 $r\sigma \overset{*}{\underset{\mathcal{IRR}(R)}{\rightarrowtail}} p_{r\sigma}$ 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 $C[\langle q, p \rangle] \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} \langle q_1, p_1 \rangle$. Then there exists a state p_2 of $\mathcal{IRR}(R)$ such that $C[\langle q, p_{r\sigma} \rangle] \overset{*}{\underset{\mathcal{B}}{\rightarrowtail}} \langle q_1, p_2 \rangle$. 50 ■

Proof.
Note that we have $p = p_1 = p_{\text{red}}$. We have to show that all the transitions used in the path $C[\langle q, p_{\text{red}} \rangle] \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} \langle q_1, p_{\text{red}} \rangle$ 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 46. □

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 $g(c) \in \mathscr{L}(\mathcal{A}_2, \left\langle q_{gfb}, p_c \right\rangle)$.

5. Correctness and precision

5.1. Proof of correctness

Definition 51 (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 $v \underset{\mathcal{A}}{\overset{*}{\rightarrowtail}} C[r\sigma]$. 51 ◀

Lemma 52.

Any automaton produced by innermost completion starting from some $\mathcal{A}_\circ \times \mathcal{IRR}(R)$ is correct wrt. R_{in}. 52 ■

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.[2] For each $i \in [\![1\,;k]\!]$, let $\langle q_i, p_i\rangle$ be the state of $\mathcal{A}$ such that $\gamma_i\mu \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} \gamma_i\sigma \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} \langle q_i, p_i\rangle$ in the path of recognition of $\ell\sigma$. Then, by consistency with $\mathcal{IRR}(R)$, for each $i \in [\![1\,;k]\!]$, $\gamma_i\mu \overset{*}{\underset{\mathcal{IRR}(R)}{\rightarrowtail}} p_i$. 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 $v_0 \in \mathcal{L}(\mathcal{A}, \langle q_0, p_0\rangle)$ and show that there is a p such that $v \in \mathcal{L}(\mathcal{A}, \langle q, p\rangle)$. 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 50. □

Theorem 53 (Correctness).

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

$$\mathcal{L}(\mathcal{A}_{\text{in}*}) \supseteq R^*_{\text{in}}(\mathcal{L}(\mathcal{A}_\circ \times \mathcal{IRR}(R))).$$ 53 ■

Proof.

Let $\mathcal{A}_{\text{in}*}$ be the calculated fixpoint automaton. By lemma 49, $\mathcal{A}_{\text{in}*}$ is consistent with $\mathcal{IRR}(R)$, and therefore, by lemma 52, $\mathcal{A}_{\text{in}*}$ is correct wrt. R_{in}. Since this automaton is a fixpoint, the case of definition 51 where there remains a critical pair cannot occur, and therefore, for all state $\langle q, p_{\text{red}}\rangle$ of $\mathcal{A}$, for all $u \in \mathcal{L}(\mathcal{A}_{\text{in}*}, \langle q, p_{\text{red}}\rangle)$ and for all $v \in R_{\text{in}}(u)$, there is a p' such that $v \in \mathcal{L}(\mathcal{A}_{\text{in}*}, \langle q, p\rangle)$. Thus, $\mathcal{A}_{\text{in}*}$ is R_{in}-closed: $R_{\text{in}}(\mathcal{L}(\mathcal{A}_{\text{in}*})) \subseteq \mathcal{L}(\mathcal{A}_{\text{in}*})$. Since the completion process only adds transitions, $\mathcal{L}(\mathcal{A}_{\text{in}*}) \supseteq \mathcal{L}(\mathcal{A}_\circ \times \mathcal{IRR}(R))$, which concludes the proof. □

5.2. Proof of precision

We will give an upper bound to the approximation induced by the equations of E. For this, we need the hypothesis that the initial automaton does not mix different equivalence classes into the same state.

Definition 54 (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 $p \in \Pi_2(Q_\mathcal{A})$, $\mathcal{L}(\mathcal{A}, \langle q, p\rangle) \subseteq [s]_E$. 54 ◀

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 55 (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 $\mathcal{L}(\mathcal{A}, \langle q, p\rangle) \neq \varnothing$, then the common E-class of all terms recognised by $\mathcal{A}$ into states of the form $\langle q, \cdot\rangle$ is noted $[q]^\mathcal{A}_E$. If there is no such p, we note $[q]^\mathcal{A}_E = \bot$.

We extend this notation to configurations of $\mathcal{A}$ as follows. First, we note $[\langle q, p\rangle]^\mathcal{A}_E = [q]^\mathcal{A}_E$. 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 $[c_i]^\mathcal{A}_E = \bot$, then let $[c]^\mathcal{A}_E = \bot$. Else, for each $i \in [\![1\,;k]\!]$, let $t_i \in \mathcal{L}(\Pi_1(\mathcal{A}), \Pi_1(c_i))$. We note $[c]^\mathcal{A}_E = [f(t_1, \ldots, t_n)]_E$. 55 ◀

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

Definition 56 (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 $\mathcal{A}$, $[c]_E^{\mathcal{A}} \neq \bot$. 56 ◀

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

Definition 57 (Reflexivity equations).
For a signature $(\Sigma, \mathcal{X})$, the set of its reflexivity equations is

$$E_r = \{f(x_1,\ldots,x_k) = f(x_1,\ldots,x_k) \mid k \in \mathbb{N} \wedge f \in \Sigma_k \wedge x_1,\ldots,x_k \text{ are distinct}\}.$$ 57 ◀

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

1. $\mathcal{A}^{\not\mathfrak{R}}$ totally separates the classes of E,
2. $\mathcal{A}$ is accessible, and
3. for any state $\langle q, p\rangle$ of $\mathcal{A}$, $\mathscr{L}(\mathcal{A}, \langle q, p\rangle) \subseteq (R_{\text{in}}/E)^* \left([q]_E^{\mathcal{A}^{\not\mathfrak{R}}}\right)$. 58 ◀

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

Lemma 59.
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 $[q_1]_E^{\mathcal{A}} = [q_2]_E^{\mathcal{A}}$. 59 ■

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 60.
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 $\mathscr{L}(\mathcal{A}, r\sigma) \subseteq (R_{\text{in}}/E)^* \left([q]_E^{\mathcal{A}^{\not\mathfrak{R}}}\right)$. 60 ■

Proof.

let $t \in T(\Sigma)$ such that $t \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} r\sigma \overset{*}{\underset{\mathcal{A}^{\not\mathfrak{K}}}{\rightarrowtail}} \langle q', p_{r\sigma} \rangle$. 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 $t \in (R_{\text{in}}/E)^* \left([q]_E^{\mathcal{A}^{\not\mathfrak{K}}} \right)$. □

This is the main theorem of preservation.

Theorem 61.

Equational simplification preserves R_{in}/E-coherence. 61 ■

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 $\mathscr{L}(\mathcal{B}^{\not\mathfrak{K}}, \langle q, p \rangle) \subseteq [q]_E^{\mathcal{A}^{\not\mathfrak{K}}}$. Consider $u = C[u_0] \overset{*}{\underset{\mathcal{A}^{\not\mathfrak{K}}}{\rightarrowtail}} C[\langle q_1, p_0 \rangle] \overset{\mathfrak{E}}{\rightarrowtail} C[\langle q_2, p_0 \rangle] \overset{*}{\underset{\mathcal{A}^{\not\mathfrak{K}}}{\rightarrowtail}} \langle q, p \rangle$. We have $u_0 \in [q_1]_E^{\mathcal{A}^{\not\mathfrak{K}}}$, thus, by Lemma 59, $u_0 \in [q_2]_E^{\mathcal{A}^{\not\mathfrak{K}}}$. Therefore $u \in C[[q_2]_E^{\mathcal{A}^{\not\mathfrak{K}}}]$, which is just $[q]_E^{\mathcal{A}^{\not\mathfrak{K}}}$. Other cases are either trivial, symmetrical or reducible to this one.

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

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

Second, assume that $C[\langle q_2, p_0 \rangle] \overset{*}{\underset{\mathcal{A}}{\rightarrowtail}} \langle q, p \rangle$ with just one $\mathfrak{K}$-transition, that is $C[\langle q_2, p_0 \rangle] \overset{*}{\underset{\mathcal{A}^{\not\mathfrak{K}}}{\rightarrowtail}} D[r\sigma] \overset{*}{\underset{\mathcal{A}^{\not\mathfrak{K}}}{\rightarrowtail}} D[\langle q_3', p_3 \rangle] \overset{\mathfrak{K}}{\rightarrowtail} D[\langle q_3, p_3 \rangle] \overset{*}{\underset{\mathcal{A}^{\not\mathfrak{K}}}{\rightarrowtail}} \langle q, p \rangle$. There is a corresponding critical pair $(\ell \to r, \sigma, \langle q_3, p_{\text{red}} \rangle)$ and, by Lemma 60, $\mathscr{L}(\mathcal{A}, r\sigma) \subseteq (R_{\text{in}}/E)^* \left([q_3]_E^{\mathcal{A}^{\not\mathfrak{K}}} \right)$. On the other hand, $D[[q_3]_E^{\mathcal{A}^{\not\mathfrak{K}}}] = [q]_E^{\mathcal{A}^{\not\mathfrak{K}}}$, so $\mathscr{L}(\mathcal{A}, D[r\sigma]) \subseteq (R_{\text{in}}/E)^* \left([q]_E^{\mathcal{A}^{\not\mathfrak{K}}} \right)$. 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 $[q]_E^{\mathcal{A}^{\not\mathfrak{K}}}$ by R_{in}/E. Since $C[[q_2]_E^{\mathcal{A}^{\not\mathfrak{K}}}] = D[[r\sigma]_E^{\mathcal{A}^{\not\mathfrak{K}}}]$, u is a descendant of such a term, so $u \in (R_{\text{in}}/E)^* \left([q]_E^{\mathcal{A}^{\not\mathfrak{K}}} \right)$.

Finally, in the paragraph above, it suffices that $D[[q_3]_E^{\mathcal{A}^{\not\mathfrak{K}}}] \subseteq (R_{\text{in}}/E)^* \left([q]_E^{\mathcal{A}^{\not\mathfrak{K}}} \right)$ (we had $D[[q_3]_E^{\mathcal{A}^{\not\mathfrak{K}}}] = [q]_E^{\mathcal{A}^{\not\mathfrak{K}}}$): this allows us to reuse this case as an induction step over the number of $\mathfrak{K}$-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 62.

Normalisation preserves R_{in}/E-coherence. So do the supplementary operations performed after completing a critical pair. 62 ■

Lemma 63.
Completion of a critical pair preserves R_{in}/E-coherence. 63 ■

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 $\langle q', p'\rangle \overset{\mathfrak{K}}{\rightarrowtail} \langle q, p'\rangle$. 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 60. □

Theorem 61 and Lemma 63 can be summed as follows.

Theorem 64.
Innermost equational completion preserves R_{in}/E-coherence. 64 ■

Theorem 65 (Precision).
Let E be a set of equations. Let $\mathcal{A}_0 = Acc(\mathcal{A}_\circ \times \mathcal{IRR}(R))$, where $\mathcal{A}_\circ$ has designated final 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

$$\mathscr{L}(\mathcal{A}) \subseteq (R_{\text{in}}/E)^*(\mathscr{L}(\mathcal{A}_\circ \times \mathcal{IRR}(R))).$$ 65 ■

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 64 show that this property is passed to $\mathcal{A}$, which yields the result. □

6. Improving accuracy of 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]. However, none of those recent static analysis techniques for functional program takes evaluation strategy into account. For instance, in Example 30, all those techniques will consider the term $c(a(s(0)), f(n))$ as reachable, though it is not with innermost strategy. As shown in Example 48, this is not the case with innermost completion.

On a simple functional program, we can show that this has an impact on the precision of the analysis. Let us consider the following Ocaml [LDG+12] program, where `sum x` computes the sum of the `x` first natural numbers, and its encoding into the TRS R:

```
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) $sum(X) \to nth(X, sumList(0, 0))$

The call-by-value strategy of OCaml corresponds to innermost rewriting[3]. We 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. First, using Timbuk[Tim12], we can compute the automaton $\mathcal{IRR}(R)$. 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:

$$\begin{tikzcd}
sum(\langle q_0, p_i\rangle) \arrow[r, "R"] \arrow[d, tail, "\mathcal{A}^0"', "*"] & nth(\langle q_0, p_i\rangle, sumList(0,0)) \arrow[d, tail, dashed, "*"', "\mathcal{A}^1"] \\
\langle q_1, p_{\text{red}}\rangle & \langle q_3, p_{\text{red}}\rangle \arrow[l, dashed, "\mathfrak{R}", "\mathcal{A}^1"']
\end{tikzcd}$$

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 46, 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 $\langle q_3, p_{\text{red}}\rangle \overset{\mathfrak{R}}{\rightarrowtail} \langle q_1, p_{\text{red}}\rangle$. 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:

$$\begin{tikzcd}
sumList(\langle q_0, p_i\rangle, \langle q_0, p_i\rangle) \arrow[r, "R"] \arrow[d, tail, "\mathcal{A}^1"', "*"] & cons(\langle q_0, p_i\rangle + \langle q_0, p_i\rangle, sumList(\langle q_0, p_i\rangle + \langle q_0, p_i\rangle, s(\langle q_0, p_i\rangle))) \arrow[d, tail, dashed, "*"', "\mathcal{A}^2"] \\
\langle q_2, p_{\text{red}}\rangle & \langle q_6, p_{\text{red}}\rangle \arrow[l, dashed, "\mathfrak{R}", "\mathcal{A}^2"']
\end{tikzcd}$$

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 $\langle q_6, p_{\text{red}}\rangle \overset{\mathfrak{R}}{\rightarrowtail} \langle q_2, p_{\text{red}}\rangle$. Note that right-hand side of

[3]More precisely, it is rightmost innermost, *i.e.* before calling a function its arguments are evaluated first and from right to left.

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 $cons(x, cons(y, z)) = cons(y, z)$ and $(x + y) + z = x + y$ is enough to have a terminating completion. Rule (3) produces infinite lists with infinite sums of the form $(((0+1)+3)+6)+\ldots$. The first equation is used to merge together all the lists and the second one is used to merge all the sums. Completion terminates on an automaton (see section A) 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.

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. Related work

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. This is not in the scope of the work presented here, though it is possible with tree automata completion in general [GS13]. Besides, we have shown that tree automata completion can take strategies into account. All aforementioned work does not take evaluation strategies into account and analysis results can thus be coarse when program execution rely on a specific strategy.

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. 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 example of Section 3 and 6 are in the scope of innermost completion but are outside of the class of [RV02]. It is due to the fact that a right-hand side of a rule has two nested defined symbols.

8. 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 $\mathcal{IRR}(R)$), instead of computing this product after each completion step.

Approximations of sets of ancestors or descendants can improve existing termination techniques [Mid02, GHWZ05] such as the dependency pairs. In particular, one can prune edges in a dependency graph using such approximations. Dependency graphs can prove innermost

termination [GTSKF06]. In this case, a precise approximation of innermost descendants should prune the innermost dependency graph, and provide a finer innermost termination criterion.

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.

For further work, we want to implement innermost completion as we have done for standard completion [Tim12] in order to design a strategy-aware 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[4].

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.

[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.

[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.

[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.

[4]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.

[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.

[LDG+12] X. Leroy, D. Doligez, J. Garrigue, D. Rémy, and J. Vouillon. The Objective Caml system release 3.12 – Documentation and user's manual. INRIA, 2012. `http://caml.inria.fr/ocaml/`.

[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.

[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.

[Tim12] Timbuk – reachability analysis and Tree Automata Calculations. IRISA / Université de Rennes 1, 2012. `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. Completed automaton for the `sum` function

Let R be the TRS:

$$
\begin{array}{l}
(1)\ 0 + X \to X \\
(2)\ s(X) + Y \to s(X + Y) \\
(3)\ sumList(X, Y) \to cons(X + Y, sumList(X + Y, s(Y))) \\
(4)\ nth(0, cons(X, Y)) \to X \\
(5)\ nth(s(X), cons(Y, Z)) \to nth(X, Z) \\
(6)\ sum(X) \to nth(X, sumList(0, 0))
\end{array}
$$

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.

A.1. First completion step

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

$$
\begin{tikzcd}
sum(\langle q_0, p_i \rangle) \arrow[r, "R"] \arrow[d, tail, "\mathcal{A}^0"', "*"] & nth(\langle q_0, p_i \rangle, sumList(0, 0)) \arrow[d, tail, dashed, "*"', "\mathcal{A}^1"] \\
\langle q_1, p_{\text{red}} \rangle & \langle q_3, p_{\text{red}} \rangle \arrow[l, dashed, tail, "\mathcal{R}", "\mathcal{A}^1"']
\end{tikzcd}
$$

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 $\langle q_3, p_{\text{red}} \rangle \overset{\mathcal{R}}{\rightarrowtail} \langle q_1, p_{\text{red}} \rangle$.

A.2. Second completion step

The second completion step solves one critical pair:

$$
\begin{tikzcd}
sumList(\langle q_0, p_i \rangle, \langle q_0, p_i \rangle) \arrow[r, "R"] \arrow[d, tail, "\mathcal{A}^1"', "*"] & cons(\langle q_0, p_i \rangle + \langle q_0, p_i \rangle, sumList(\langle q_0, p_i \rangle + \langle q_0, p_i \rangle, s(\langle q_0, p_i \rangle))) \arrow[d, tail, dashed, "*"', "\mathcal{A}^2"] \\
\langle q_2, p_{\text{red}} \rangle & \langle q_6, p_{\text{red}} \rangle \arrow[l, dashed, tail, "\mathcal{R}", "\mathcal{A}^2"']
\end{tikzcd}
$$

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 $\langle q_6, p_{\text{red}} \rangle \overset{\mathcal{R}}{\rightarrowtail} \langle q_2, p_{\text{red}} \rangle$.

A.3. Third completion step

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

$$
\begin{tikzcd}
0 + \langle q_0, p_i \rangle \arrow[r, "R"] \arrow[d, tail, "\mathcal{A}^2"', "*"] & \langle q_0, p_i \rangle \arrow[dl, dashed, tail, "\mathfrak{R}", "\mathcal{A}^3"'] \\
\langle q_4, p_{\mathrm{red}} \rangle & \\
\langle q_4, p_i \rangle &
\end{tikzcd}
\qquad
\begin{tikzcd}
s(\langle q_0, p_i \rangle) + \langle q_0, p_i \rangle \arrow[r, "R"] \arrow[d, tail, "\mathcal{A}^2"', "*"] & s(\langle q_0, p_i \rangle + \langle q_0, p_i \rangle) \arrow[d, tail, dashed, "*"', "\mathcal{A}^3"] \\
\langle q_4, p_{\mathrm{red}} \rangle & \langle q_7, p_{\mathrm{red}} \rangle \arrow[l, dashed, tail, "\mathfrak{R}", "\mathcal{A}^3"']
\end{tikzcd}
$$

For solving the second critical pair, since the transition $\langle q_0, p_i \rangle + \langle q_0, p_i \rangle \rightarrowtail \langle q_4, p_{\mathrm{red}} \rangle$ already belong to $\mathcal{A}_2$, it is enough to add the transitions: $s(\langle q_4, p_{\mathrm{red}} \rangle) \rightarrowtail \langle q_7, p_{\mathrm{red}} \rangle$ and $\langle q_7, p_{\mathrm{red}} \rangle \overset{\mathfrak{R}}{\rightarrowtail} \langle q_4, p_{\mathrm{red}} \rangle$. For solving the first critical pair, it is enough to add the transition $\langle q_0, p_i \rangle \overset{\mathfrak{R}}{\rightarrowtail} \langle q_4, p_i \rangle$ and to add supplementary transitions, as mentioned in Definition 46. Because of the separation between states $\langle q_4, p_{\mathrm{red}} \rangle$ and $\langle q_4, p_i \rangle$, it is necessary to add a copy of all transitions with a state $\langle q_4, p_{\mathrm{red}} \rangle$ in their left-hand side where we replace it by state $\langle q_4, p_i \rangle$ to . We thus add the transitions: $cons(\langle q_4, p_i \rangle, \langle q_5, p_{\mathrm{red}} \rangle) \rightarrowtail \langle q_6, p_{\mathrm{red}} \rangle$, $sumList(\langle q_4, p_i \rangle, \langle q_0, p_i \rangle) \rightarrowtail \langle q_5, p_{\mathrm{red}} \rangle$ and $s(\langle q_4, p_i \rangle) \rightarrowtail \langle q_7, p_i \rangle$.

A.4. Fourth critical pair

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

$$
\begin{tikzcd}
sumList(\langle q_4, p_i \rangle, \langle q_0, p_i \rangle) \arrow[r, "R"] \arrow[d, tail, "\mathcal{A}^3"', "*"] & cons(\langle q_4, p_i \rangle + \langle q_0, p_i \rangle, sumList(\langle q_4, p_i \rangle + \langle q_0, p_i \rangle, s(\langle q_0, p_i \rangle))) \arrow[d, tail, dashed, "*"', "\mathcal{A}^4"] \\
\langle q_5, p_{\mathrm{red}} \rangle & \langle q_8, p_{\mathrm{red}} \rangle \arrow[l, dashed, tail, "\mathfrak{R}", "\mathcal{A}^4"']
\end{tikzcd}
$$

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

$$
\begin{tikzcd}
(\langle q_0, p_i \rangle + \langle q_0, p_i \rangle) + \langle q_0, p_i \rangle \arrow[r, equal, "E"] \arrow[d, tail, "\mathcal{A}"', "*"] & \langle q_0, p_i \rangle + \langle q_0, p_i \rangle \arrow[d, tail, "*"', "\mathcal{A}"] \\
\langle q_9, p_{\mathrm{red}} \rangle \arrow[r, dashed, tail, "\mathfrak{E}"] & \langle q_4, p_{\mathrm{red}} \rangle \arrow[l, dashed, tail, "\mathfrak{E}"]
\end{tikzcd}
$$

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:

$$\begin{tikzcd}
cons(\langle q_4, p_{red}\rangle, cons(\langle q_9, p_{\mathrm{red}}\rangle, \langle q_{10}, p_{\mathrm{red}}\rangle)) \arrow[r, equal, "E"] \arrow[d, tail, "\mathcal{A}"', "*"] & cons(\langle q_9, p_{\mathrm{red}}\rangle, \langle q_{10}, p_{\mathrm{red}}\rangle) \arrow[d, tail, "*"', "\mathcal{A}"] \\
\langle q_6, p_{\mathrm{red}}\rangle \arrow[r, tail, dashed, shift left, "\mathfrak{E}"] & \langle q_8, p_{\mathrm{red}}\rangle \arrow[l, tail, dashed, shift left, "\mathfrak{E}"]
\end{tikzcd}$$

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

A.5. Fith completion step

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

$$\begin{tikzcd}
0 + \langle q_0, p_i\rangle \arrow[r, "R"] \arrow[d, tail, "\mathcal{A}^4"', "*"] & \langle q_0, p_i\rangle \arrow[d, tail, "\mathfrak{R}"', "\mathcal{A}^4"] \\
\langle q_9, p_{\mathrm{red}}\rangle & \langle q_4, p_{\mathrm{red}}\rangle \arrow[dl, tail, "\mathfrak{E}", "\mathcal{A}^4"'] \\
\langle q_9, p_i\rangle &
\end{tikzcd}$$

$$\begin{tikzcd}
s(\langle q_0, p_i\rangle) + \langle q_0, p_i\rangle \arrow[rr, "R"] \arrow[d, tail, "\mathcal{A}^4"', "*"] & & s(\langle q_0, p_i\rangle + \langle q_0, p_i\rangle) \arrow[d, tail, "*"', "\mathcal{A}^4"] \\
\langle q_9, p_{\mathrm{red}}\rangle & \langle q_4, p_i\rangle \arrow[l, tail, "\mathfrak{E}", "\mathcal{A}^4"'] & \langle q_7, p_{\mathrm{red}}\rangle \arrow[l, tail, "\mathfrak{R}", "\mathcal{A}^4"']
\end{tikzcd}$$

There remains a last critical pair which is:

$$\begin{tikzcd}
sumList(\langle q_9, p_i\rangle, \langle q_0, p_i\rangle) \arrow[r, "R"] \arrow[d, tail, "\mathcal{A}^4"', "*"] & cons(\langle q_9, p_i\rangle + \langle q_0, p_i\rangle, sumList(\langle q_9, p_i\rangle + \langle q_0, p_i\rangle, s(\langle q_0, p_i\rangle))) \arrow[d, tail, dashed, "*"', "\mathcal{A}^5"] \\
\langle q_{10}, p_{\mathrm{red}}\rangle & \langle q_8, p_{\mathrm{red}}\rangle \arrow[l, tail, dashed, "\mathfrak{R}", "\mathcal{A}^5"']
\end{tikzcd}$$

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, $\langle q_6, p_{\mathrm{red}}\rangle \overset{\mathfrak{E}}{\rightarrowtail} \langle q_8, p_{\mathrm{red}}\rangle$. Finally the only transition to add is thus $\langle q_8, p_{\mathrm{red}}\rangle \overset{\mathfrak{R}}{\rightarrowtail} \langle q_{10}, p_{\mathrm{red}}\rangle$. Since no new transition is added on symbol $sumList$, this ends the completion. All remaining critical pairs are joinable, in particular:

$$
\begin{tikzcd}
sumList(\langle q_9, p_i\rangle, \langle q_0, p_i\rangle) \arrow[r, "R"] \arrow[d, "\mathcal{A}^4"', "*"] & cons(\langle q_9, p_i\rangle + \langle q_0, p_i\rangle, sumList(\langle q_9, p_i\rangle + \langle q_0, p_i\rangle, s(\langle q_0, p_i\rangle))) \arrow[d, "*"', "\mathcal{A}^4"] \\
\langle q_{10}, p_{\mathrm{red}}\rangle & \langle q_8, p_{\mathrm{red}}\rangle \arrow[l, "\mathfrak{R}", "\mathcal{A}^4"']
\end{tikzcd}
$$

The transitions of the completed automaton are:

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

$\langle q_6, p\rangle \overset{\mathfrak{E}}{\rightleftarrows} \langle q_8, p\rangle$ for all p

$\langle q_8, p_{\mathrm{red}}\rangle \overset{\mathfrak{R}}{\rightarrowtail} \langle q_{10}, p_{\mathrm{red}}\rangle$

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{\mathfrak{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{\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_4, p_i\rangle, \langle q_{10}, p_{\mathrm{red}}\rangle)))) \\
&\overset{\mathfrak{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{\mathfrak{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{\mathfrak{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{\mathfrak{R}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, \langle q_6, p_{\mathrm{red}}\rangle)) \\
&\overset{\mathfrak{E}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, \langle q_8, p_{\mathrm{red}}\rangle)) \\
&\overset{\mathfrak{R}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, cons(\langle q_0, p_i\rangle, \langle q_5, p_{\mathrm{red}}\rangle)) \\
&\overset{\mathfrak{R}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, cons(\langle q_4, p_i\rangle, \langle q_5, p_{\mathrm{red}}\rangle)) \\
&\overset{\mathfrak{R}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, \langle q_6, p_{\mathrm{red}}\rangle) \\
&\overset{\mathfrak{E}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, \langle q_8, p_{\mathrm{red}}\rangle) \\
&\overset{\mathfrak{R}}{\rightarrowtail} cons(\langle q_0, p_i\rangle, \langle q_5, p_{\mathrm{red}}\rangle) \\
&\overset{\mathfrak{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{\mathfrak{R}}{\rightarrowtail} \langle q_2, p_{\mathrm{red}}\rangle
\end{aligned}
$$