A Condensed Goal-Independent Fixpoint Semantics Modeling the Small-Step Behavior of Rewriting

Marco Comini[1] and Luca Torella[2]

[1] DIMI, University of Udine, Italy — marco.comini@uniud.it
[2] DIISM, University of Siena, Italy — luca.torella@unisi.it

Abstract

In this paper we present a novel condensed narrowing-like semantics that contains the minimal information which is needed to describe compositionally all possible rewritings of a term rewriting system. We provide its goal-dependent top-down definition and, more importantly, an equivalent goal-independent bottom-up fixpoint characterization.

We prove soundness and completeness w.r.t. the small-step behavior of rewriting for the *full class* of term rewriting systems.

1 Introduction

Nowadays the formalism of Term Rewriting Systems (TRSs) is used, besides for functional programming, also for many other applications (like specification of communication protocols, to mention one). There has been a lot of research on the development of tools for the formal verification and (in general) automatic treatment/manipulation of TRSs. Within the proposals there are semantics-based approaches which can guarantee correctness by construction. However they cannot employ directly the construction of the semantics, since in general it is infinite. Thus some kind of approximation has to be used.

Given the potentiality of application of the TRS formalism, we have turned our attention toward the development of semantics-based TRS manipulation tools with the intention to use Abstract Interpretation theory as fundament to devise semantics approximations correct by construction. However, as also noted by [9], defining a suitable (concrete) semantics is usually the first crucial step in adapting the general methodology of Abstract Interpretation to the semantic framework of the programming language at hand. When a concrete semantics is used to define, via abstract interpretation, abstract (approximate) semantics to be employed to develop semantics-based manipulation tools, it is *particularly* relevant if it is *condensed* and defined compositionally. In the literature, a semantics is said to be *condensing* when the semantics of an instance of an expression (term, goal, call) can be obtained with a semantic operation directly from the semantics of the (un-instantiated) expression. In such a situation, only the semantics for most general expressions can be maintained in denotations. We say that a semantics is *condensed* when the denotations themselves do not contain redundancy, i.e., when it is not possible to semantically derive the components of a denotation from the other components. Indeed, the abstract semantics operations which are obtained from a condensed concrete semantics involve the use of the join operation (of the abstract domain) at each iteration in parallel onto all components of rules, instead of using several subsequent applications for all components. This has a twofold benefit. On one side, it speeds up convergence of the abstract fixpoint computation. On the other side, it considerably improves precision.

In [2], we developed an automatic debugging methodology for the TRS formalism based on abstract interpretation of the big-step rewriting semantics that is most commonly considered in functional programming, i.e., the set of constructor terms/normal forms. However, the resulting tool was inefficient. The main reason for this inefficiency is because the chosen concrete semantics is not condensing and thus, because of its accidental high redundancy, it causes the algorithms to use and produce much redundant information at each stage. In contrast, the same methodology gave good results in [5] because it was applied to a condensed semantics.

The constructor terms/normal forms semantics is not condensed because it contains all possible rewritings, but there are many possible rewritings which can be obtained by some other ones, since rewriting is closed under substitution (stability) and replacement ($t \underset{\mathcal{R}}{\longrightarrow}^{*} s$ implies $C[t]_p \underset{\mathcal{R}}{\longrightarrow}^{*} C[s]_p$). Thus, in [1] we tried to directly devise a semantics, fully abstract w.r.t. the big-step rewriting semantics, with the specific objective to avoid all redundancy while still characterizing the rewritings of any term. In particular we searched a semantics which

- has a compositional goal-independent definition,
- is the fixpoint of a bottom-up construction,
- is as condensed as possible.

Unfortunately (in [1]) we just partially achieved this goal since the semantics is defined only for some classes of TRSs. In the meantime, for a (quite) different language, in [4, 3] we obtained—for the full language—a semantics with the mentioned characteristics, by following a different approach:

1. Define a denotation, *fully abstract* w.r.t. the *small-step* behavior of evaluation of expressions, which enjoys the mentioned properties.
2. Obtain by abstraction of this small-step semantics a denotation (which enjoys the mentioned properties) correct w.r.t. the *big-step* behavior.

This approach has the additional advantage that the small-step semantics can be reused also to develop other semantics more concrete than the *big-step* one (for instance semantics which can model functional dependencies that are suitable to develop pre-post verification methods).

Unfortunately in the case of the TRS formalism we do not have a suitable *small-step* semantics to start with. For Curry we defined the small-step semantics by collecting just the most general traces of the small-step operational semantics, which correspond (in our case) to the rewriting derivations of the terms $f(x_1, \dots, x_n)$. The problem is that we cannot obtain, just from the traces of all $f(x_1, \dots, x_n)$, the rewriting derivations of all (nested) terms, without using again (directly or indirectly) the rewriting mechanism. In fact, usually $f(x_1, \dots, x_n)$ is immediately a normal form, because we cannot instantiate variables; however, there are many instances which can trigger the rules. *Narrowing* [7] can seem a possible solution to this problem but we have an issue related to the interference of non-confluence (i.e., non-determinism) with non-linearity, as shown by this example.

Example 1.1

Let us consider the following TRS $\mathcal{R}$:

$$\begin{array}{lll} \mathit{coin} \to \mathit{Tail} & \mathit{Head} \neq \mathit{Tail} \to \mathit{True} & \mathit{diff}(x) \to x \neq x \\ \mathit{coin} \to \mathit{Head} & \mathit{Tail} \neq \mathit{Head} \to \mathit{True} & \end{array}$$

We have rewriting derivations $\mathit{diff}(x) \to x \neq x \not\to$, $\mathit{diff}(\mathit{Head}) \to \mathit{Head} \neq \mathit{Head} \not\to$, $\mathit{diff}(\mathit{Tail}) \to \mathit{Tail} \neq \mathit{Tail} \not\to$, while $\mathit{diff}(\mathit{coin}) \to^{*} \mathit{True}$. Moreover, we have the narrowing derivation $\mathit{diff}(x) \overset{\varepsilon}{\rightsquigarrow} x \neq x \not\rightsquigarrow$.

Narrowing can instantiate variables (according to rules), but a variable is instantiated with the same term in all of its occurrences (it would make little sense to do differently, for a top-down resolution mechanism). However, Example 1.1 shows that it is not possible to retrieve that $\mathit{diff}(\mathit{coin}) \to^* \mathit{True}$ from all possible narrowing derivations of $\mathit{diff}(x)$, since the only narrowing derivation (of $\mathit{diff}(x)$) does not reach True.

In this paper we define a variation of narrowing (linearizing narrowing) which admits different instances of variables with multiple occurrences. With linearizing narrowing we define a denotation, *fully abstract* w.r.t. the *small-step* behavior of rewriting, which enjoys the mentioned properties *for generic TRSs without restrictions.* The outline to achieve this is the following.

- We gather all linearizing narrowing derivations into trees (Definition 3.10).
- We show that all possible rewritings can be reconstructed from linearizing narrowing trees (Theorem 3.14).
- We define top-down condensed denotations $\mathcal{O}[\![\mathcal{R}]\!]$ by collecting just the linearizing narrowing trees of most general terms ($f(\overrightarrow{x_n})$) and we prove that, with a suitable semantic evaluation function $\mathcal{E}$, we can reconstruct any linearizing narrowing tree starting from $\mathcal{O}[\![\mathcal{R}]\!]$ (Theorem 3.23).
- By using $\mathcal{E}$ we define a (bottom-up) immediate consequence operator whose least fixpoint $\mathcal{F}[\![\mathcal{R}]\!]$ is equal to $\mathcal{O}[\![\mathcal{R}]\!]$ (Theorem 3.30). Thus from $\mathcal{F}[\![\mathcal{R}]\!]$ we can reconstruct all possible rewritings of $\mathcal{R}$ and we have full abstraction w.r.t. the rewriting behavior (Corollary 3.31).

Note that the proofs of all results are in the appendix.

2 Preliminaries

We assume that the reader is familiar with the basic notions of term rewriting. For a thorough discussion of these topics, see [10]. In the paper we use the following notions and notations.

We write $\overrightarrow{o_n}$ for the *list* of syntactic objects $o_1, \dots, o_n$. Given a monotonic function $F : \mathcal{L} \to \mathcal{L}$, over lattice $\mathcal{L}$ whose bottom is $\perp$ and *lub* is $\bigsqcup$, by $F{\uparrow}k$ we denote function $\lambda x. F^k(x)$ and by $F{\uparrow}\omega$ function $\lambda x. \bigsqcup\{F^k(x) \mid k \in \mathbb{N}\}$. By $\mathit{lfp}(F)$ we denote the least fixed point of F (and recall that, for a continuos F, $\mathit{lfp}(F) = F{\uparrow}\omega$).

Terms and Substitutions

Σ denotes a signature and $\mathcal{V}$ denotes a (fixed) countably infinite set of variables. $\mathcal{T}(\Sigma, \mathcal{V})$ denotes the terms built over signature Σ and variables $\mathcal{V}$. Σ is *partitioned* in $\mathcal{D}$, the *defined* symbols (also called operations), and $\mathcal{C}$, the *constructor* symbols (also called data constructors). $\mathcal{T}(\mathcal{C}, \mathcal{V})$ are called *constructor terms.* The set of variables occurring in a term t is denoted by $\mathit{var}(t)$, while the sequence (in order) of variables occurring in a term t is denoted by $\overrightarrow{\mathit{var}}(t)$. A term is *linear* if it does not contain multiple occurrences of any variable. $\mathcal{LT}(\Sigma, \mathcal{V})$ denotes the set of linear terms.

$t|_p$ denotes the *subterm* of t at position p, and $t[s]_p$ denotes the result of *replacing the subterm* $t|_p$ by the term s.

Given a substitution $\vartheta = \{x_1/t_1, \dots, x_n/t_n\}$ we denote by $\mathit{dom}(\sigma)$ and $\mathit{range}(\sigma)$ the domain set $\{x_1, \dots, x_n\}$ and the range set $\bigcup_{i=1}^{n} \mathit{var}(t_i)$ respectively. The identity substitution is denoted by ε. By $t\sigma$ we denote the application of σ to t. $\sigma{\restriction}_V$ denotes the restriction of substitution σ to set $V \subseteq \mathcal{V}$. $\sigma\vartheta$ denotes the composition of ϑ and σ i.e., the substitution s.t. $x(\sigma\vartheta) = (x\sigma)\vartheta$ for any $x \in \mathcal{V}$. Given two substitutions ϑ_1 and ϑ_2 and two terms t_1 and t_2, we say that ϑ_1

(respectively t_1) is more general than ϑ_2 (respectively t_2), denoted $\vartheta_1 \leq \vartheta_2$ (respectively $t_1 \leq t_2$) if and only if there exists a substitution σ s.t. $\vartheta_1\sigma = \vartheta_2$ (respectively $t_1\sigma = t_2$). We denote by $\simeq$ the induced equivalence, i.e., $\vartheta \simeq \sigma$ if and only if there exists a renaming ρ s.t. $\vartheta\rho = \sigma$ (and $\sigma\rho^{-1} = \vartheta$). With $\sigma \uparrow \sigma'$ we indicate the *lub* (w.r.t. $\leq$) of σ and σ'.

A substitution ϑ is an *unifier* for terms t and s if $t\vartheta = s\vartheta$. t and s *unify/are unifiable* when there exists a unifier. A unifier σ for t and s is a *most general unifier*, denoted as $\sigma = mgu(t, s)$, when $\sigma \leq \vartheta$ for any unifier ϑ of t and s.

Rewriting

A *term rewriting system* (TRS for short) $\mathcal{R}$ is a set of rules $l \to r$ where $l, r \in \mathcal{T}(\Sigma, \mathcal{V})$, $var(r) \subseteq var(l)$, $l = f(t_1, \dots, t_n)$ and $f \in \mathcal{D}$. $t_1, \dots, t_n$ are the argument patterns of $l \to r$ and need not necessarily be in $\mathcal{T}(\mathcal{C}, \mathcal{V})$, unlike in functional programming, where only *constructor-based* TRSs are considered (i.e., with $t_i \in \mathcal{T}(\mathcal{C}, \mathcal{V})$). We denote by $\mathbb{R}_\Sigma$ the set of all TRSs defined on signature Σ.

Given TRS $\mathcal{R}$, a rewrite step $t \xrightarrow[\mathcal{R}]{p} t'$ is defined if there are a position p in t, $l \to r \in \mathcal{R}$ and a substitution η with $dom(\eta) \subseteq var(l)$ such that $t|_p = l\eta$ and $t' = t[r\eta]_p$. As usual, we omit to write position p when it is not relevant and omit $\mathcal{R}$ when is clear from the context. Moreover we use $\to^*$ to denote the transitive and reflexive closure of the rewriting relation $\to$.

A term t is called a *normal form*, denoted by $t \not\to$, if there is no term s such that $t \to^* s$.

A substitution $\{x_1/t_1, \dots, x_n/t_n\}$ is $\mathcal{R}$-normalized (w.r.t. a TRS $\mathcal{R}$) if all t_i are normal forms (which trivially includes the case when $t_i \in \mathcal{T}(\mathcal{C}, \mathcal{V})$).

Full Narrowing

In the paper with $s \ll X$ we denote a *renamed apart* variant s of an element belonging to a set of syntactic objects X, i.e., a renaming of variable names of some $x \in X$ that does not contain variables that appear in the context of the definition where $s \ll X$ is used (this is also called "using fresh variables names in s").

The combination of variable instantiation and rewriting is called *narrowing* [7]. Formally, a (full) narrowing step $t \underset{\mathcal{R}}{\overset{\sigma,p}{\rightsquigarrow}} t'$ is defined if there is a position p in t, $l \to r \ll \mathcal{R}$ and $\sigma = mgu(t|_p, l)$ such that $t|_p \notin \mathcal{V}$ and $t' = (t[r]_p)\sigma$. In such a case we have that $t\sigma \xrightarrow[\mathcal{R}]{p} t'$. Again, we omit to write position p when it is not relevant and omit $\mathcal{R}$ when is clear from the context.

$t \not\rightsquigarrow$ denotes that there is no term s such that $t \rightsquigarrow s$.

3 Modeling the small-step rewriting behavior

In this section we introduce the concrete semantics which is suitable to model the small-step rewriting behavior. In order to formally state such relationship we first need to formally define the concept of small-step rewriting behavior.

Definition 3.1 (Rewriting behavior) *Given* $t_0 \in \mathcal{T}(\Sigma, \mathcal{V})$ *and* $\mathcal{R} \in \mathbb{R}_\Sigma$ *the* small-step rewriting behavior of t_0 in $\mathcal{R}$ *is*

$$\mathcal{B}^{ss}[\![t_0 \text{ } in \text{ } \mathcal{R}]\!] := \left\{ t_0 \xrightarrow[\mathcal{R}]{} t_1 \xrightarrow[\mathcal{R}]{} \cdots \xrightarrow[\mathcal{R}]{} t_{n-1} \xrightarrow[\mathcal{R}]{} t_n \;\middle|\; \forall t_i \in \mathcal{T}(\Sigma, \mathcal{V}) \right\} \tag{3.1}$$

and the small-step rewriting behavior of $\mathcal{R}$ *is* $\mathcal{B}^{ss}[\![\mathcal{R}]\!] := \bigcup_{t \in \mathcal{T}(\Sigma, \mathcal{V})} \mathcal{B}^{ss}[\![t \text{ } in \text{ } \mathcal{R}]\!]$.

This notion of observable behavior induces the definition of TRS equivalence:

$$\forall \mathcal{R}_1, \mathcal{R}_2 \in \mathbb{R}_\Sigma. \mathcal{R}_1 \approx_{ss} \mathcal{R}_2 :\Longleftrightarrow \mathcal{B}^{ss}[\![\mathcal{R}_1]\!] = \mathcal{B}^{ss}[\![\mathcal{R}_2]\!] \tag{3.2}$$

Thanks to the following property we can restrain the check of $\mathcal{B}^{ss}$ equivalence of two TRSs to linear terms. Moreover in the sequel we will also restrict our attention only to denotations for linear terms.

Proposition 3.2 *Let $\mathcal{R}_1, \mathcal{R}_2 \in \mathbb{R}_\Sigma$. Then $\mathcal{R}_1 \approx_{ss} \mathcal{R}_2 \iff \mathcal{R}_1$ is a variant of $\mathcal{R}_2 \iff \forall t \in \mathcal{LT}(\Sigma, \mathcal{V})$. $\mathcal{B}^{ss}[\![t \textit{ in } \mathcal{R}_1]\!] = \mathcal{B}^{ss}[\![t \textit{ in } \mathcal{R}_2]\!]$.*

3.1 The semantic domain

We now define a notion of "hypothetical rewriting" which resembles full narrowing [7], but it "decouples" multiple occurrences of variables. This way we maintain the potentiality to choose different evolutions of the rewritings of redexes once variables are instantiated with non constructor terms.

The semantic domain of our semantics will be made of trees with all possible derivations of this variation of narrowing.

3.1.1 Linearizing Narrowing

Definition 3.3 (Term Linearization) *Let $t \in \mathcal{T}(\Sigma, \mathcal{V})$, $r \in \mathcal{LT}(\Sigma, \mathcal{V})$ and $\sigma : \mathcal{V} \to \mathcal{V}$. We say that (r, σ) is a* linearization *of t, denoted $(r, \sigma) = lin(t)$, if $r\sigma = t$ and $var(t) \subseteq var(r)$. The substitution σ will be called* delinearizator*.*

If a term is linear then $lin(t) = (t, \varepsilon)$, while for non-linear terms we can have different possibilities (for instance $lin(f(x,x)) = (f(x,y), \{y/x\}) = (f(z,x), \{z/x\}) = \ldots$). However the following constructions which involve linearization are actually independent upon the particular choice of linearization, in the sense that all possible different results are variants (analogously to what happens for the choice of different *mgu*'s).

It may be useful to note that a delinearizer σ has one binding for each (further) multiple occurrence of a variable.

Definition 3.4 (Linearizing Narrowing Derivation) *Let $t, s \in \mathcal{LT}(\Sigma, \mathcal{V})$ and $\mathcal{R} \in \mathbb{R}_\Sigma$. There exists a* linearizing narrowing step *$t \overset{\theta,p}{\underset{\sigma,\mathcal{R}}{\Longrightarrow}} s$ if there exist a position p of t, $l \to r \ll \mathcal{R}$, $\theta' = mgu(t|_p, l)$ and $\sigma : \mathcal{V} \to \mathcal{V}$ such that*

$$t|_p \notin \mathcal{V}, \qquad (s, \sigma) = lin(t[r\theta']_p), \qquad \theta = \theta'\restriction_{var(t)}.$$

We omit to write position p when it is not relevant and omit $\mathcal{R}$ when is clear from the context.

A sequence $t_0 \overset{\theta_1}{\underset{\sigma_1}{\Longrightarrow}} t_1 \ldots \overset{\theta_n}{\underset{\sigma_n}{\Longrightarrow}} t_n$ is called linearizing narrowing derivation*. With $t_0 \overset{\theta}{\underset{\sigma}{\Longrightarrow}}{}^* t_n$ we denote the existence of a linearizing narrowing derivation such that $\theta = \theta_1 \cdots \theta_n$ and $\sigma = \sigma_1 \cdots \sigma_n$.*

Note that in linearizing narrowing derivations we do not apply *mgu* θ to all reduct $(t[r]_p)\theta$ as narrowing does. This would not make any difference since terms are kept linear by construction and thus θ cannot alter the context outside positions being reduced.

Linearizing narrowing is correct w.r.t. rewriting (as narrowing is) as proven by the following theorem which is the analogous of Theorem 3.8 of [8] (Theorem 1 of [7]) where we use linearizing narrowing instead of narrowing. Actually linearizing narrowing is more general than narrowing, in the sense that when we have a narrowing derivation then we will surely have a linearizing narrowing derivation which possibly compute more general instances (in case there are non-linear terms).

Theorem 3.5 *Let $\mathcal{R} \in \mathbb{R}_\Sigma$, $s_0 \in \mathcal{LT}(\Sigma, \mathcal{V})$, $t_0 \in \mathcal{T}(\Sigma, \mathcal{V})$, η_0 an $\mathcal{R}$-normalized substitution such that $t_0 = s_0\eta_0$ and $V \subset \mathcal{V}$ such that $var(s_0) \cup dom(\eta_0) \subseteq V$. If $t_0 \to^* t_n$ then there exist a term $s_n \in \mathcal{LT}(\Sigma, \mathcal{V})$ and substitutions η_n, θ, σ such that*

$$s_0 \overset{\theta}{\underset{\sigma}{\Longrightarrow}}{}^* s_n, \qquad t_n = s_n\sigma\eta_n, \qquad (\theta\eta_n){\restriction_V} = \eta_0{\restriction_V}, \qquad \eta_n \textit{ is } \mathcal{R}\textit{-normalized},$$

where $s_0 \overset{\theta}{\underset{\sigma}{\Longrightarrow}}{}^* s_n$ and $t_0 \to^* t_n$ employ the same rewrite rules at the same positions.

Note that while it is always possible to transform rewrite derivations into linearizing narrowing derivations (in the sense of Theorem 3.5), the opposite does not hold in general as we will show in Example 3.7. As anticipated, we gather all linearizing narrowing derivations (of the same term t) into a tree.

Definition 3.6 (Linearizing Narrowing Trees) *Let $t \in \mathcal{LT}(\Sigma, \mathcal{V})$. A* linearizing narrowing tree *T for t is a (not necessarily finite) labelled tree which is rooted in t and where*

1. *paths are linearizing narrowing derivations;*
2. *sibling subtrees have the same root terms if and only if their incoming arcs have different substitutions.*

We denote with $\mathbb{LNT}_\Sigma$ (or simply $\mathbb{LNT}$ when clear from the context) the set of all the linearizing narrowing trees (over Σ). Moreover, for any $t \in \mathcal{LT}(\Sigma, \mathcal{V})$, we denote with $\mathbb{LNT}_t$ the set of all linearizing narrowing trees for t.

Point 2 ensures that all sibling steps in a linearizing tree are pairwise distinct and thus that we cannot have two different paths of the tree with the same terms and labels.

Example 3.7

Consider TRS $\mathcal{R}$ of Example 1.1. The linearizing narrowing tree starting from term $\mathit{diff}(x)$ is:

$$\begin{tikzcd}
 & & \mathit{True} \\
\mathit{diff}(x) \arrow[r, Rightarrow, "\varepsilon", "\{x_1/x\}"'] & x \neq x_1 \arrow[ru, Rightarrow, "\{x/\mathit{Tail}, x_1/\mathit{Head}\}", "\varepsilon"'] \arrow[rd, Rightarrow, "\{x/\mathit{Head}, x_1/\mathit{Tail}\}", "\varepsilon"'] & \\
 & & \mathit{True}
\end{tikzcd}$$

This linearizing narrowing derivation $\mathit{diff}(x) \overset{\{x/\mathit{Head}, x_1/\mathit{Tail}\}}{\underset{\{x_1/x\}}{\Longrightarrow}}{}^* \mathit{True}$ can be read as: if there is a term t that can rewrite both to Head and Tail, then $\mathit{diff}(t)$ rewrites to True. Indeed $\mathit{diff}(\mathit{coin})$ does rewrite to True (a possible rewriting derivation is $\mathit{diff}(\mathit{coin}) \to \mathit{coin} \neq \mathit{coin} \to \mathit{Head} \neq \mathit{coin} \to \mathit{Head} \neq \mathit{Tail} \to \mathit{True}$).

In this case is not possible to transform these linearizing narrowing derivations into rewrite derivations (since $\mathit{diff}(x) \to x \neq x \not\to$, as well as $\mathit{diff}(t) \to t \neq t \not\to$, for all $t \in \mathcal{T}(\mathcal{C}, \mathcal{V})$).

This example also shows that linearizing narrowing can have longer derivations w.r.t. (standard) narrowing since $\mathit{diff}(x) \overset{\varepsilon}{\leadsto} x \neq x \not\leadsto$.

Definition 3.8 (Variance on $\mathbb{LNT}$) *Let $t \in \mathcal{LT}(\Sigma, \mathcal{V})$ and $T_1, T_2 \in \mathbb{LNT}_t$. We say that T_1 and T_2 are* local variants *if there exists a renaming ρ, such that $T_1 \rho = T_2$.*

Two linearizing narrowing trees are local variants if and only if they have the same root t and their steps are equal up to renaming of variables which do not occur in t.

Note: Since the actual choices of local variable names is completely irrelevant, from now on, with an abuse of notation, by $\mathbb{LNT}$ we will actually indicate its quotient w.r.t. local variance. Moreover all linearizing narrowing trees presented in the sequel will actually be an arbitrary representative of an equivalence class.

Definition 3.9 (Order on $\mathbb{LNT}$) *Let denote with $paths(T)$ the set of all the paths of T starting from the root.*

Given $T_1, T_2 \in \mathbb{LNT}$, we define $T_1 \sqsubseteq T_2$ if and only if $paths(T_1) \subseteq paths(T_2)$.

Given a set $\mathcal{T} \subseteq \mathbb{LNT}_t$, the least upper bound $\bigsqcup \mathcal{T}$ is the tree whose paths are $\bigcup_{T \in \mathcal{T}} paths(T)$. Dually for the greatest lower bound $\bigsqcap$.

It is worth noticing that, for any $t \in \mathcal{LT}(\Sigma, \mathcal{V})$, $\mathbb{LNT}_t$ is a complete lattice.

By Point 2 of Definition 3.6, *paths* is injective, thus it establishes an order preserving isomorphism $(\mathbb{LNT}_t, \sqsubseteq) \underset{paths}{\overset{prfxtree}{\leftleftarrows\!\!\!\rightarrow}} (paths(\mathbb{LNT}_t), \subseteq)$, where the adjoint of *paths*, *prfxtree*, builds a tree from a set of paths (by merging all common prefixes). So we have two isomorphic representations of linearizing narrowing trees and in the sequel we can simply write $d \in T$ for $d \in paths(T)$. The set representation is very convenient for technical definitions, while for examples the tree representation is better suited.

Definition 3.10 (Linearizing Narrowing Tree of a term) *Let $t \in \mathcal{LT}(\Sigma, \mathcal{V})$ and $\mathcal{R} \in \mathbb{R}_\Sigma$. A* linearizing narrowing tree *$\mathcal{N}[\![t \text{ in } \mathcal{R}]\!]$ for term t in TRS $\mathcal{R}$ is*

$$\mathcal{N}[\![t \text{ in } \mathcal{R}]\!] := \{d \in \mathbb{LNT}_t \mid d \text{ uses rules from } \mathcal{R}\}$$

Intuitively, $\mathcal{N}[\![t \text{ in } \mathcal{R}]\!]$ denotes the linearizing narrowing behavior of linear term t in TRS $\mathcal{R}$ modulo local variance (i.e., local variables are up to renaming).

Example 3.11

Given the following TRS $\mathcal{R}$:

$$m(H(x)) \to d(x, K(x)) \qquad d(C(x), K(E(y))) \to f(x, x, y, y)$$
$$f(A, x, y, F) \to B(y, y) \qquad f(E(x), y, A, A) \to K(E(y))$$

The linearizing narrowing tree $\mathcal{N}[\![m(x) \text{ in } \mathcal{R}]\!]$ is:

$$\begin{tikzcd}
 & & & K(E(x_4)) \\
m(x) \arrow[r, Rightarrow, "\{x/H(x_1)\}", "\{x_2/x_1\}"'] & d(x_1, K(x_2)) \arrow[r, Rightarrow, "{\{x_1/C(x_3), x_2/E(y)\}}", "{\{x_4/x_3, y_1/y\}}"'] & f(x_3, x_4, y, y_1) \arrow[ur, Rightarrow, "{\{x_3/E(x_5), y/A, y_1/A\}}", "\varepsilon"'] \arrow[dr, Rightarrow, "{\{x_3/A, y_1/F\}}", "\{y_2/y\}"'] & \\
 & & & B(y, y_2)
\end{tikzcd}$$

Linearizing narrowing trees "capture" the behavioral TRS equivalence $\approx_{ss}$ since the TRS equivalence induced by $\mathcal{N}$ coincides with $\approx_{ss}$.

Theorem 3.12 *Let $\mathcal{R}_1, \mathcal{R}_2 \in \mathbb{R}_\Sigma$. Then $\mathcal{R}_1 \approx_{ss} \mathcal{R}_2$ if and only if, for every $t \in \mathcal{LT}(\Sigma, \mathcal{V})$, $\mathcal{N}[\![t \text{ in } \mathcal{R}_1]\!] = \mathcal{N}[\![t \text{ in } \mathcal{R}_2]\!]$.*

Even more, from linearizing narrowing trees we can completely reconstruct the small-step rewriting behavior. To formally prove this we need to introduce the following operation to build rewriting derivations from linearizing narrowing derivations.

Definition 3.13 *Let $\mathcal{R} \in \mathbb{R}_\Sigma$ and $d = s_0 \underset{\sigma_1}{\overset{\theta_1}{\Longrightarrow}} s_1 \underset{\sigma_2}{\overset{\theta_2}{\Longrightarrow}} \dots \underset{\sigma_n}{\overset{\theta_n}{\Longrightarrow}} s_n$ be a linearizing narrowing derivation. The* linear narrowing to rewriting operator *is defined as*

$$\lfloor d \rfloor := \left\{ t_0 \to t_1 \dots \to t_k \;\middle|\; \begin{array}{l} \xi \text{ is an } \mathcal{R}\text{-normalized substitution},\ 0 \leq k \leq n, \\ \eta = \theta_1 \uparrow \sigma_1 \uparrow \dots \uparrow \theta_k \uparrow \sigma_k,\ \forall 0 \leq i \leq k.\, t_i = s_i \eta \xi \end{array} \right\} \tag{3.3}$$

We abuse notation and lift $\lfloor \cdot \rfloor$ also to sets as $\lfloor S \rfloor := \bigcup_{d \in S} \lfloor d \rfloor$.

Intuitively, this operation takes a prefix d of a linear narrowing derivation and, if it can simultaneously satisfy all its computed answers and delinearizators, with substitution η, then it builds a rewriting sequence by applying $\eta\xi$ to all terms of d, for any $\mathcal{R}$-normalized substitution ξ.

Theorem 3.14 *Let $\mathcal{R} \in \mathbb{R}_\Sigma$ and $t \in \mathcal{LT}(\Sigma, \mathcal{V})$. Then $\mathcal{B}^{ss}[\![t \text{ in } \mathcal{R}]\!] = \lfloor \mathcal{N}[\![t \text{ in } \mathcal{R}]\!] \rfloor$.*

Hence $\mathcal{N}[\![t \text{ in } \mathcal{R}]\!]$ is indeed a condensed representation of $\mathcal{B}^{ss}[\![t \text{ in } \mathcal{R}]\!]$. Now the next step for the construction of a semantics with the desired characteristics is to achieve compositionality. To do so we should look for a denotation for most general terms $f(\overrightarrow{x_n})$ (of a TRS $\mathcal{R}$) which could be used to retrieve, with suitable semantic operations, $\mathcal{N}[\![t \text{ in } \mathcal{R}]\!]$ for any $t \in \mathcal{LT}(\Sigma, \mathcal{V})$.

3.2 Operational denotations of TRSs

The operational denotation of a TRS can be defined as an interpretation giving meaning to the defined symbols over linearizing narrowing trees "modulo variance". Essentially we define the semantics of each function in $\mathcal{D}$ over formal parameters (whose names are actually irrelevant).

Definition 3.15 (Interpretations) *Let $\mathbb{MGT}_\mathcal{D} := \{ f(\overrightarrow{x_n}) \mid f_{/n} \in \mathcal{D}, \overrightarrow{x_n}$ are distinct variables$\}$.*

Two functions $I, J \colon \mathbb{MGT}_\mathcal{D} \to \mathbb{LNT}_\Sigma$ are (global) variants*, denoted by $I \cong J$, if for each $\pi \in \mathbb{MGT}_\mathcal{D}$ there exists a renaming ρ such that $(I(\pi))\rho = J(\pi\rho)$.*

An interpretation *is a function $\mathcal{I} \colon \mathbb{MGT}_\mathcal{D} \to \mathbb{LNT}_\Sigma$ modulo variance*[2] *such that, for every $\pi \in \mathbb{MGT}_\mathcal{D}$, $\mathcal{I}(\pi)$ is a linearizing narrowing tree for π.*

The semantic domain $\mathbb{I}_\Sigma$ (or simply $\mathbb{I}$ when clear from the context) is the set of all interpretations ordered by the pointwise extension of $\sqsubseteq$.

The partial order on $\mathbb{I}$ formalizes the evolution of the computation process. $(\mathbb{I}, \sqsubseteq)$ is a complete lattice and its least upper bound and greatest lower bound are the pointwise extension of $\bigsqcup$ and $\bigsqcap$, respectively. In the sequel we abuse the notations for $\mathbb{LNT}$ for $\mathbb{I}$ as well. The bottom element of $\mathbb{I}$ is $\bot_\mathbb{I} := \lambda\pi.\{\pi\}$ (for each $\pi \in \mathbb{MGT}_\mathcal{D}$). In the sequel we abuse the notations for $\mathbb{LNT}$ for $\mathbb{I}$ as well.

It is important to note that $\mathbb{MGT}_\mathcal{D}$ (modulo variance) has the same cardinality of $\mathcal{D}$ (and is then finite) and thus each interpretation is a finite collection (of possibly infinite elements). Hence we will often explicitly write interpretations by cases, like

$$\mathcal{I} := \begin{cases} \pi_1 \mapsto T_1 \\ \vdots \\ \pi_n \mapsto T_n \end{cases} \quad \text{for} \quad \begin{array}{c} \mathcal{I}(\pi_1) := T_1 \\ \vdots \\ \mathcal{I}(\pi_n) := T_n \end{array}$$

[2]i.e., a family of elements of $\mathbb{LNT}_\Sigma$, indexed by $\mathbb{MGT}_\mathcal{D}$, modulo variance.

In the following, any $\mathcal{I} \in \mathbb{I}$ is implicitly considered as an arbitrary function $\mathbb{MGT} \to \mathbb{LNT}$ obtained by choosing an arbitrary representative of the elements of $\mathcal{I}$ in the equivalence class generated by $\cong$. Actually, in the sequel, all the operators that we use on $\mathbb{I}$ are also independent of the choice of the representative. Therefore, we can define any operator on $\mathbb{I}$ in terms of its counterpart defined on functions $\mathbb{MGT} \to \mathbb{LNT}$.

Moreover, we also implicitly assume that the application of an interpretation $\mathcal{I}$ to a specific $\pi \in \mathbb{MGT}$, denoted by $\mathcal{I}(\pi)$, is the application $I(\pi)$ of any representative I of $\mathcal{I}$ which is defined exactly on π. For example if $\mathcal{I} = (\lambda f(x,y).\, f(x,y) \xRightarrow[\varepsilon]{\{x/c(z)\}} c(y,z))\big/_{\cong}$ then $\mathcal{I}(f(u,v)) = f(u,v) \xRightarrow[\varepsilon]{\{u/c(z)\}} c(v,z)$.

While defined symbols have to be interpreted according to TRS rules, constructor symbols are meant to be interpreted as themselves. In order to treat them as a generic case of function application, we assume that *any* interpretation $\mathcal{I}$ is also implicitly extended on constructors as $\mathcal{I}(c(\overrightarrow{x_n})) \coloneqq c(\overrightarrow{x_n})$. In the sequel we will use φ when we refer to a generic (either constructor or defined) symbol, whence f for defined symbols and c for constructor ones.

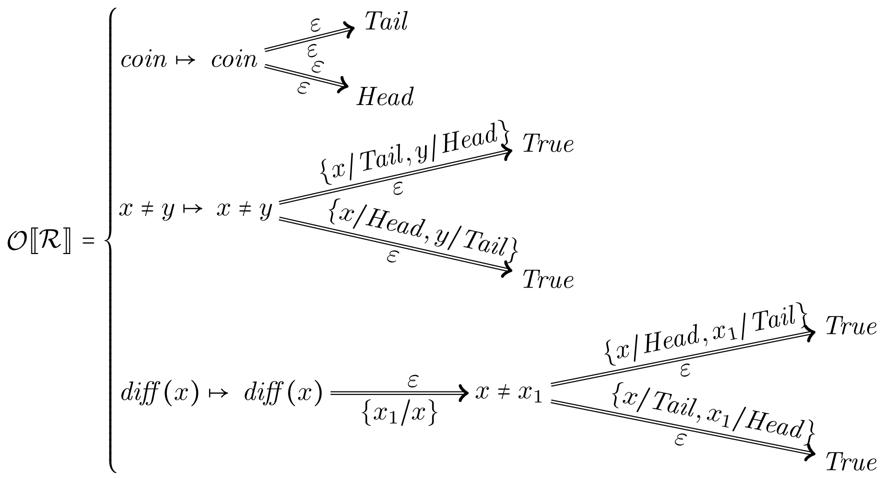
Definition 3.16 (Operational denotation of TRSs) *Let $\mathcal{R} \in \mathbb{R}_\Sigma$. Then the* operational denotation *of $\mathcal{R}$ is*

$$\mathcal{O}[\![\mathcal{R}]\!] \coloneqq \big(\lambda f(\overrightarrow{x_n}).\, \mathcal{N}[\![f(\overrightarrow{x_n}) \text{ in } \mathcal{R}]\!]\big)\big/_{\cong} \tag{3.4}$$

Intuitively, $\mathcal{O}$ collects the linearizing narrowing tree of each $f(\overrightarrow{x_n})$ in $\mathcal{R}$, abstracting from the particular choice of the variable names $\overrightarrow{x_n}$.

Example 3.17

The operational denotation of TRS $\mathcal{R}$ of Example 1.1 is:



$$\mathcal{O}[\![\mathcal{R}]\!] = \left\{ \begin{array}{l}
\begin{tikzcd} & & \mathit{Tail} \\ \mathit{coin} \mapsto & \mathit{coin} \arrow[ur, Rightarrow, "\varepsilon", "\varepsilon"'] \arrow[dr, Rightarrow, "\varepsilon", "\varepsilon"'] & \\ & & \mathit{Head} \end{tikzcd} \\
\begin{tikzcd} & & \mathit{True} \\ x \neq y \mapsto & x \neq y \arrow[ur, Rightarrow, "\{x/\mathit{Tail}, y/\mathit{Head}\}", "\varepsilon"'] \arrow[dr, Rightarrow, "\{x/\mathit{Head}, y/\mathit{Tail}\}", "\varepsilon"'] & \\ & & \mathit{True} \end{tikzcd} \\
\begin{tikzcd} & & & \mathit{True} \\ \mathit{diff}(x) \mapsto & \mathit{diff}(x) \arrow[r, Rightarrow, "\varepsilon", "\{x_1/x\}"'] & x \neq x_1 \arrow[ur, Rightarrow, "\{x/\mathit{Head}, x_1/\mathit{Tail}\}", "\varepsilon"'] \arrow[dr, Rightarrow, "\{x/\mathit{Tail}, x_1/\mathit{Head}\}", "\varepsilon"'] & \\ & & & \mathit{True} \end{tikzcd}
\end{array} \right.$$

The (small-step) rewriting behavior of any term t can be "reconstructed" from $\mathcal{O}[\![\mathcal{R}]\!]$ by means of the following *evaluation function* $\mathcal{E}$.

3.2.1 Evaluation Function

When we have the interpretation with the linearizing narrowing tree of $f(\overrightarrow{x_n})$, we can easily reconstruct the rewriting behavior of any $f(\overrightarrow{t_n})$ for (renamed apart) $\overrightarrow{t_n} \in \mathcal{LT}(\mathcal{C}, \mathcal{V})$, by simply replacing the most general bindings along a derivation with constructor terms t_i (clearly pruning branches with inconsistent instances). However, with non-constructor nested terms things gets more involved. In practice we have an interleaving of parts of all sub-derivations corresponding to the evaluation of arguments, leaded by the derivation of $f(\overrightarrow{x_n})$. Intuitively the basic building block of our proposal is the definition of a semantics embedding operation that mimics parameter passing. Namely taken two linearizing narrowing trees T_1, T_2 and a variable x of (the root of) T_1, the tree-embedding operation $T_1[x/T_2]$ transforms T_1 by modifying its steps accordingly to steps of T_2, which provides specific actual parameter values to x in places where x in T_1 was originally "freely" instantiated.

In order to define $T_1[x/T_2]$ we need to introduce an auxiliary (technical) relation which works on single derivations. Note that in the sequel, to shorten definitions, *when* we adorn linearizing narrowing derivations or trees with a term s, like d_s or T_s, we mean that the head of d_s or the root of T_s is term s.

Definition 3.18 *Let $\pi \in \mathcal{LT}(\Sigma, \mathcal{V})$, d_g be a linearizing narrowing derivation with head g and $T_b \in \mathbb{LNT}_b$. Then $d_g; \pi; T_b \vdash d$ is the least relation that satisfies the rules:*

$$\frac{}{d_t; \pi; T_s \vdash t\mu} \; \mu = \overleftarrow{mgu}(s, \pi) \tag{3.5a}$$

$$\frac{d_t; \pi; T_{s'} \vdash d}{d_t; \pi; T_s \vdash t\mu \overset{\theta,q}{\underset{\sigma}{\Longrightarrow}} d} \; \mu = \overleftarrow{mgu}(s, \pi) \;,\; s \overset{\theta,p}{\underset{\sigma}{\Longrightarrow}} T_{s'} \in T_s,\; \exists q \,.\, s|_p = t\mu|_q \tag{3.5b}$$

$$\frac{d_{t_0}; \pi_0; T_{s_0} \vdash d_{t_1} \;\ldots\; d_{t_n}; \pi_n; T_{s_n} \vdash d_{t_{n+1}}}{(t \overset{\theta,p}{\underset{\sigma}{\Longrightarrow}} d_{t_0}); \pi; T_{s_0} \vdash t\mu \overset{\theta',p}{\underset{\sigma''}{\Longrightarrow}} d_{t_{n+1}}} \; \begin{array}{l} \mu = \overleftarrow{mgu}(s_0, \pi), \\ \theta' = \overleftarrow{mgu}(t\mu, t\theta)\restriction_{var(t\mu)} \;, \\ \{x_1/y_1, \ldots, x_n/y_n\} = proj(var(\pi\theta), \sigma), \\ T_{s_1}, \ldots, T_{s_n} \ll T_{s_0}, \\ \pi_0 = \pi\theta,\; \forall i \in \{1, \ldots, n\}\; \pi_i = \pi_0\{y_i/x_i\}, \\ \forall j \in \{0, \ldots, n\}\; \overrightarrow{v_j} = \overrightarrow{var}(s_j), \\ \sigma'' = (\sigma \cup \bigcup_{i=1}^{n} \overrightarrow{v_i}/\overrightarrow{v_0})\restriction_{var(t_{n+1})} \end{array} \tag{3.5c}$$

where

- *$proj(V, \sigma) = \{x/y \in \sigma \mid y \in V\}$,*
- *$\overleftarrow{mgu}(t, s)$ is an mgu θ of t and s such that $\forall x \in var(t)\; x\theta \notin \mathcal{V}$.*

Broadly speaking, the role of π in a statement $d_t; \pi; T_s \vdash d$ is that of the "parameter pattern" responsible to constrain "freely" instantiated formal parameters in d_t to the actual parameters values which are actually "coming" from T_s. More specifically, Rules 3.5 govern the inlaying of the steps of a linearizing narrowing tree T_s into a derivation d_t. In particular

- The axiom 3.5a stops any further possible inlaying.
- The rule 3.5b considers the case when we want to proceed with an inner linearizing narrowing step employing a step coming from T_s. In this case t plays the role of context and the inner step is done accordingly to the one chosen from T_s. Note that is it possible to do the step only if exists q such that $s|_p = t\mu|_q$. Namely the defined symbol, over which the step is done, needs to be "visible" in $t\mu$.
- The rule 3.5c considers the case when we want to do an outermost step. First, the step has to be compatible with the way we instantiated t so far, namely exists $mgu(t\mu, t\theta)$. Then, we choose from σ only the delinearizers that depend on the variable from which we started the embedding. Each of these delinearizer comes from the linearization of a multiple occurrence of a same variable z. Thus, for each rename z' of z we sequentially embed into z' a (possibly different) sub-derivation coming from a renamed apart variant of

T_s. Note that if we would not have multiple occurrences of z (i.e., $proj(var(\pi\theta),\sigma) = \varnothing$), the rule would simply be

$$\frac{d_{t_0};\pi_0;T_{s_0} \vdash d_{t_1}}{(t \xRightarrow[\sigma]{\theta,p} d_{t_0});\pi;T_{s_0} \vdash t\mu \xRightarrow[\sigma]{\theta',p} d_{t_1}} \quad \begin{array}{l}\mu = \overleftarrow{mgu}(s_0,\pi)\ ,\\ \theta' = \overleftarrow{mgu}(t\mu,t\theta)\restriction_{var(t\mu)}\ ,\\ \pi_0 = \pi\theta\end{array}$$

Note that in Rules 3.5 we use a selected form of mgu (i.e., $\overleftarrow{mgu}$ which does not rename variables in the left argument) in order to avoid to change variable names along the way.

Example 3.19

Consider $\mathcal{O}[\![\mathcal{R}]\!]$ of Example 3.17. Let $T_{coin} \coloneqq \mathcal{O}[\![\mathcal{R}]\!](coin)$ and $d_0 \coloneqq \mathit{diff}(x) \xRightarrow[\{x_1/x\}]{\varepsilon} d_1 \in \mathcal{O}[\![\mathcal{R}]\!](\mathit{diff}(x))$, where $d_1 = x \neq x_1 \xRightarrow[\varepsilon]{\{x/Head,x_1/Tail\}} True$.

Let us build a proof tree to find a derivation d such that $d_0;x;T_{coin} \vdash d$. We have to start with an application of rule (3.5c) which (in this case) has two premises since the delinearizator $\{x_1/x\}$ in the first step of d_0 has one binding. The first subtree which embeds the $Head$ branch of T_{coin} into x is

$$\text{(3.5b)}\ \frac{\text{(3.5c)}\ \dfrac{\text{(3.5a)}\ \dfrac{}{True;Head;Head \vdash True}}{d_1;x;Head \vdash Head \neq x_1 \xRightarrow[\varepsilon]{\{x_1/Tail\}} True}}{d_1;x;T_{coin} \vdash \underbrace{coin \neq x_1 \xRightarrow[\varepsilon]{\varepsilon} Head \neq x_1 \xRightarrow[\varepsilon]{\{x_1/Tail\}} True}_{d_3}} \tag{PT}$$

Now we can build the full proof tree (by building the second subtree which, starting from d_3, can finish to embed the $Tail$ branch of T_{coin} into x_1).

$$\text{(3.5c)}\ \frac{\text{(PT)} \qquad \text{(3.5c)}\ \dfrac{\text{(3.5b)}\ \dfrac{\text{(3.5c)}\ \dfrac{\text{(3.5a)}\ \dfrac{}{True;\,Tail;\,Tail \vdash True}}{d_2;x_1;\,Tail \vdash Head \neq Tail \xRightarrow[\varepsilon]{\varepsilon} True}}{d_2;x_1;T_{coin} \vdash Head \neq coin \xRightarrow[\varepsilon]{\varepsilon} Head \neq Tail \xRightarrow[\varepsilon]{\varepsilon} True}}{d_3;x_1;T_{coin} \vdash coin \neq coin \xRightarrow[\varepsilon]{\varepsilon} Head \neq coin \xRightarrow[\varepsilon]{\varepsilon} Head \neq Tail \xRightarrow[\varepsilon]{\varepsilon} True}}{d;x;T_{coin} \vdash \mathit{diff}(coin) \xRightarrow[\varepsilon]{\varepsilon} coin \neq coin \xRightarrow[\varepsilon]{\varepsilon} Head \neq coin \xRightarrow[\varepsilon]{\varepsilon} Head \neq Tail \xRightarrow[\varepsilon]{\varepsilon} True}$$

We have analogous proof tree for $d;x;T_{coin} \vdash \mathit{diff}(coin) \xRightarrow[\varepsilon]{\varepsilon} coin \neq coin \xRightarrow[\varepsilon]{\varepsilon} Tail \neq coin \xRightarrow[\varepsilon]{\varepsilon} Tail \neq Head \xRightarrow[\varepsilon]{\varepsilon} True$. In total we have ten possible proof trees, four of whom that have derivations which reach $True$.

Tree-embedding is simply defined by collecting all contributes of $d;x;T \vdash d'$.

Definition 3.20 (Tree-embedding) *Let $T_g \in \mathbb{LNT}_g$ and $T_b \in \mathbb{LNT}_g$ such that*

1. *T_g and T_b do not share any local variable;*
2. *x is a variable which does not occur in T_b.*

Then the tree-embedding *operation $T_g[x/T_b]$ is defined as $T_g[x/T_b] \coloneqq \{d \mid d_g \in T_g, d_g;x;T_b \vdash d\}$.*

The evaluation function is obtained by repeated application of tree-embedding.

Definition 3.21 (Evaluation Function) *Let $t \in \mathcal{LT}(\Sigma, \mathcal{V})$ and $\mathcal{I} \in \mathbb{I}$. The evaluation of t w.r.t. $\mathcal{I}$, denoted $\mathcal{E}[\![t]\!]_{\mathcal{I}}$, is defined by induction on the structure of t as follows:*

$$\mathcal{E}[\![x]\!]_{\mathcal{I}} \coloneqq x \tag{3.6a}$$

$$\mathcal{E}[\![\varphi(\overrightarrow{t_n})]\!]_{\mathcal{I}} \coloneqq \mathcal{I}(\varphi(\overrightarrow{x_n}))[x_1/\mathcal{E}[\![t_1]\!]_{\mathcal{I}}]\dots[x_n/\mathcal{E}[\![t_n]\!]_{\mathcal{I}}] \qquad \overrightarrow{x_n} \text{ renamed apart distinct} \tag{3.6b}$$

Example 3.22

Consider $\mathcal{O}[\![\mathcal{R}]\!]$ of Example 3.17. The evaluation of $\mathcal{E}[\![\mathit{diff}(\mathit{coin})]\!]_{\mathcal{O}[\![\mathcal{R}]\!]}$ is

$$\begin{aligned}
&\mathcal{E}[\![\mathit{diff}(\mathit{coin})]\!]_{\mathcal{O}[\![\mathcal{R}]\!]} = && [\text{by Equation (3.6b) with } n = 1]\\
&\mathcal{O}[\![\mathcal{R}]\!](\mathit{diff}(x))[x/\mathcal{E}[\![\mathit{coin}]\!]_{\mathcal{O}[\![\mathcal{R}]\!]}] = && [\text{by Equation (3.6b) with } n = 0]\\
&\mathcal{O}[\![\mathcal{R}]\!](\mathit{diff}(x))[x/\mathcal{O}[\![\mathcal{R}]\!](\mathit{coin})]
\end{aligned}$$

Then, by completing what shown in Example 3.19, we have that $\mathcal{E}[\![\mathit{diff}(\mathit{coin})]\!]_{\mathcal{O}[\![\mathcal{R}]\!]}$ is

$$\begin{tikzcd}
& \mathit{Tail} \neq \mathit{Tail} & & & \mathit{Tail} \neq \mathit{Tail} & \\
& & & \mathit{coin} \neq \mathit{Tail} \arrow[ur, Rightarrow, "\varepsilon", "\varepsilon"'] \arrow[dr, Rightarrow, "\varepsilon", "\varepsilon"'] & & \\
& \mathit{diff}(\mathit{Tail}) \arrow[uu, Rightarrow, "\varepsilon", "\varepsilon"'] & & & \mathit{Head} \neq \mathit{Tail} \arrow[r, Rightarrow, "\varepsilon", "\varepsilon"'] & \mathit{True} \\
& & & & \mathit{Tail} \neq \mathit{Head} \arrow[r, Rightarrow, "\varepsilon", "\varepsilon"'] & \mathit{True} \\
& & & \mathit{coin} \neq \mathit{Head} \arrow[ur, Rightarrow, "\varepsilon", "\varepsilon"'] \arrow[dr, Rightarrow, "\varepsilon", "\varepsilon"'] & & \\
& & & & \mathit{Head} \neq \mathit{Head} & \\
\mathit{diff}(\mathit{coin}) \arrow[uuuur, Rightarrow, "\varepsilon", "\varepsilon"'] \arrow[rr, Rightarrow, "\varepsilon", "\varepsilon"'] \arrow[ddddr, Rightarrow, "\varepsilon", "\varepsilon"'] & & \mathit{coin} \neq \mathit{coin} \arrow[uuuuur, Rightarrow, "\varepsilon", "\varepsilon"'] \arrow[uur, Rightarrow, "\varepsilon", "\varepsilon"'] \arrow[ddr, Rightarrow, "\varepsilon", "\varepsilon"'] \arrow[dddddr, Rightarrow, "\varepsilon", "\varepsilon"'] & & & \\
& & & & \mathit{Tail} \neq \mathit{Tail} & \\
& & & \mathit{Tail} \neq \mathit{coin} \arrow[ur, Rightarrow, "\varepsilon", "\varepsilon"'] \arrow[dr, Rightarrow, "\varepsilon", "\varepsilon"'] & & \\
& & & & \mathit{Tail} \neq \mathit{Head} \arrow[r, Rightarrow, "\varepsilon", "\varepsilon"'] & \mathit{True} \\
& \mathit{diff}(\mathit{Head}) \arrow[dd, Rightarrow, "\varepsilon", "\varepsilon"'] & & & \mathit{Head} \neq \mathit{Tail} \arrow[r, Rightarrow, "\varepsilon", "\varepsilon"'] & \mathit{True} \\
& & & \mathit{Head} \neq \mathit{coin} \arrow[ur, Rightarrow, "\varepsilon", "\varepsilon"'] \arrow[dr, Rightarrow, "\varepsilon", "\varepsilon"'] & & \\
& \mathit{Head} \neq \mathit{Head} & & & \mathit{Head} \neq \mathit{Head} &
\end{tikzcd}$$

3.2.2 Properties of the TRS operational denotation

The following result states formally that from $\mathcal{O}[\![\mathcal{R}]\!]$ the evaluation function $\mathcal{E}$ can reconstruct the linearizing narrowing tree of any linear term.

Theorem 3.23 *For all $\mathcal{R} \in \mathbb{R}_\Sigma$ and $t \in \mathcal{LT}(\Sigma, \mathcal{V})$, $\mathcal{E}[\![t]\!]_{\mathcal{O}[\![\mathcal{R}]\!]} = \mathcal{N}[\![t \text{ in } \mathcal{R}]\!]$.*

A straightforward consequence of Theorems 3.23 and 3.12 is

Corollary 3.24 *For all $\mathcal{R}_1, \mathcal{R}_2 \in \mathbb{R}_\Sigma$, $\mathcal{O}[\![\mathcal{R}_1]\!] = \mathcal{O}[\![\mathcal{R}_2]\!]$ if and only if $\mathcal{R}_1 \approx_{ss} \mathcal{R}_2$.*

Thus semantics $\mathcal{O}$ is fully abstract w.r.t. $\approx_{ss}$.

3.3 Fixpoint denotations of TRSs

We will now define a bottom-up goal-independent denotation which is equivalent to $\mathcal{O}$ and thus (by Corollary 3.24) adequate to characterize the small-step behavior for TRSs. It is defined as the fixpoint of an abstract immediate operator over interpretations $\mathcal{P}[\![\mathcal{R}]\!]$. This operator is essentially given in terms of evaluation $\mathcal{E}$ of right hand sides of rules. Namely, given an interpretation $\mathcal{I}$, it essentially consists in:

- building an initial linearizing narrowing step for a most general term according to rules' left hand side;
- applying the evaluation operator $\mathcal{E}$ to the right hand side of the rule over $\mathcal{I}$.

Definition 3.25 *Let $\mathcal{R} \in \mathbb{R}_\Sigma$. $\mathcal{P}[\![\mathcal{R}]\!] : \mathbb{I} \to \mathbb{I}$ is defined, for all $f \in \mathcal{D}$ (of arity n), as*

$$\mathcal{P}[\![\mathcal{R}]\!]_{\mathcal{I}}(f(\overrightarrow{x_n})) := \bigsqcup \left\{ f(\overrightarrow{x_n}) \overset{\theta}{\underset{\sigma}{\Longrightarrow}} \mathcal{E}[\![r']\!]_{\mathcal{I}} \;\middle|\; \begin{array}{l} f(\overrightarrow{x_n})\theta \to r \ll \mathcal{R}, \\ (r', \sigma) = lin(r) \end{array} \right\} \tag{3.7}$$

Moreover we define our fixpoint semantics as $\mathcal{F}[\![\mathcal{R}]\!] := lfp\, \mathcal{P}[\![\mathcal{R}]\!]$.

$\mathcal{F}[\![\mathcal{R}]\!]$ is well defined since $\mathcal{P}[\![\mathcal{R}]\!]$ is continuous.

Proposition 3.26 *Let $\mathcal{R} \in \mathbb{R}_\Sigma$. Then $\mathcal{P}[\![\mathcal{R}]\!]$ is continuous.*

Example 3.27

Let us consider the (artificial) TRS $\mathcal{R} := \{g \to f(h(a)),\ h(a) \to h(b),\ f(h(b)) \to a\}$ taken from [1], which is neither constructor-based nor confluent. The evaluation of the right hand sides of all rules is $\mathcal{E}[\![f(h(a))]\!]_{\bot_{\mathbb{I}}} = \bot_{\mathbb{I}}(f(x))[x/\mathcal{E}[\![h(a)]\!]_{\bot_{\mathbb{I}}}] = f(x)[x/h(a)] = f(h(a))$; $\mathcal{E}[\![h(b)]\!]_{\bot_{\mathbb{I}}} = h(x)[x/b] = h(b)$ and $\mathcal{E}[\![a]\!]_{\bot_{\mathbb{I}}} = a$. Hence

$$\mathcal{I}_1 := \mathcal{P}[\![\mathcal{R}]\!]\!\uparrow\!1 = \begin{cases} g \mapsto\ g \xRightarrow[\varepsilon]{\varepsilon} f(h(a)) \\ h(x) \mapsto\ h(x) \xRightarrow[\varepsilon]{\{x/a\}} h(b) \\ f(x) \mapsto\ f(x) \xRightarrow[\varepsilon]{\{x/h(b)\}} a \end{cases}$$

In the next iteration we have to evaluate $\mathcal{E}[\![f(h(a))]\!]_{\mathcal{I}_1} = \mathcal{I}_1(f(x))[x/\mathcal{E}[\![h(a)]\!]_{\mathcal{I}_1}]$. Since $\mathcal{E}[\![h(a)]\!]_{\mathcal{I}_1} = \mathcal{I}_1(h(x))[x/\mathcal{E}[\![a]\!]_{\mathcal{I}_1}] = h(a) \overset{\varepsilon}{\underset{\varepsilon}{\Longrightarrow}} h(b)$ we have

$$\mathcal{P}[\![\mathcal{R}]\!]\!\uparrow\!2 = \begin{cases} g \mapsto\ g \xRightarrow[\varepsilon]{\varepsilon} f(h(a)) \xRightarrow[\varepsilon]{\varepsilon} f(h(b)) \xRightarrow[\varepsilon]{\varepsilon} a \\ h(x) \mapsto\ h(x) \xRightarrow[\varepsilon]{\{x/a\}} h(b) \\ f(x) \mapsto\ f(x) \xRightarrow[\varepsilon]{\{x/h(b)\}} a \end{cases}$$

Now, since $\mathcal{P}[\![\mathcal{R}]\!]\!\uparrow\!3 = \mathcal{P}[\![\mathcal{R}]\!]\!\uparrow\!2$, this is also the fixpoint $\mathcal{F}[\![\mathcal{R}]\!]$.

Example 3.28

Consider TRS $\mathcal{R}$ of Example 1.1. The iterates of $\mathcal{P}[\![\mathcal{R}]\!]$ are

$$\mathcal{P}[\![\mathcal{R}]\!]\!\uparrow\!1 = \begin{cases} \begin{tikzcd} & Tail \\ coin \mapsto\ coin \arrow[ur, Rightarrow, "\varepsilon", "\varepsilon"'] \arrow[dr, Rightarrow, "\varepsilon", "\varepsilon"'] & \\ & Head \end{tikzcd} \\ \begin{tikzcd} & True \\ x \neq y \mapsto\ x \neq y \arrow[ur, Rightarrow, "{\{x/Tail, y/Head\}}", "\varepsilon"'] \arrow[dr, Rightarrow, "{\{x/Head, y/Tail\}}", "\varepsilon"'] & \\ & True \end{tikzcd} \\ diff(x) \mapsto\ diff(x) \xRightarrow[\{x_1/x\}]{\varepsilon} x \neq x_1 \end{cases}$$

$$\mathcal{P}[\![\mathcal{R}]\!]{\uparrow}2 = \left\{\begin{array}{l}
\begin{tikzcd} \mathit{coin} \mapsto \mathit{coin} \arrow[r, Rightarrow, "\varepsilon", "\varepsilon"'] \arrow[dr, Rightarrow, "\varepsilon", "\varepsilon"'] & \mathit{Tail} \\ & \mathit{Head} \end{tikzcd} \\
\begin{tikzcd} x \neq y \mapsto x \neq y \arrow[r, Rightarrow, "\{x/\mathit{Tail},\, y/\mathit{Head}\}", "\varepsilon"'] \arrow[dr, Rightarrow, "\{x/\mathit{Head},\, y/\mathit{Tail}\}", "\varepsilon"'] & \mathit{True} \\ & \mathit{True} \end{tikzcd} \\
\begin{tikzcd} \mathit{diff}(x) \mapsto \mathit{diff}(x) \arrow[r, Rightarrow, "\varepsilon", "\{x_1/x\}"'] & x \neq x_1 \arrow[r, Rightarrow, "\{x/\mathit{Head},\, x_1/\mathit{Tail}\}", "\varepsilon"'] \arrow[dr, Rightarrow, "\{x/\mathit{Tail},\, x_1/\mathit{Head}\}", "\varepsilon"'] & \mathit{True} \\ & & \mathit{True} \end{tikzcd}
\end{array}\right.$$

Now, since $\mathcal{P}[\![\mathcal{R}]\!]{\uparrow}3 = \mathcal{P}[\![\mathcal{R}]\!]{\uparrow}2$, this is also the fixpoint $\mathcal{F}[\![\mathcal{R}]\!]$.

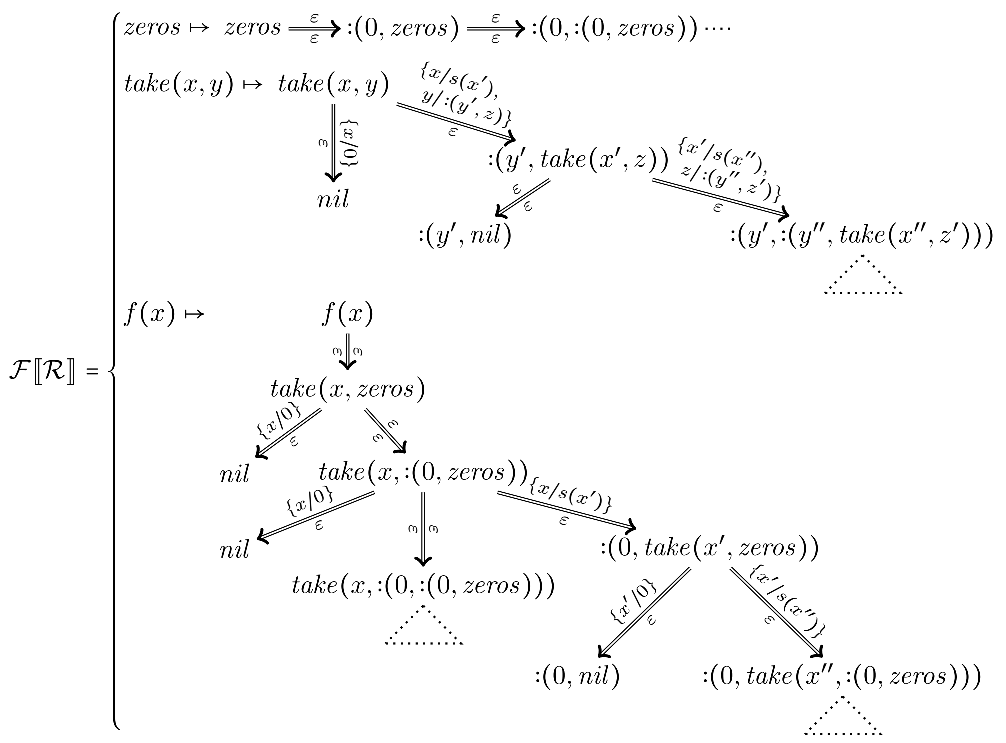
Example 3.29

Let us consider the TRS $\mathcal{R} := \{\mathit{zeros} \to {:}(0, \mathit{zeros}),\ \mathit{take}(0, x) \to \mathit{nil},\ \mathit{take}(s(n), {:}(y, z)) \to {:}(y, \mathit{take}(n, z)),\ f(n) \to \mathit{take}(n, \mathit{zeros})\}$. The first two iterates of $\mathcal{P}[\![\mathcal{R}]\!]$ are

$$\mathcal{P}[\![\mathcal{R}]\!]{\uparrow}1 = \left\{\begin{array}{l}
\begin{tikzcd} \mathit{zeros} \mapsto \mathit{zeros} \arrow[r, Rightarrow, "\varepsilon", "\varepsilon"'] & {:}(0, \boxed{\mathit{zeros}}) \end{tikzcd} \\
\begin{tikzcd} \mathit{take}(x_1, y_1) \mapsto \mathit{take}(x_1, y_1) \arrow[d, Rightarrow, "\varepsilon"', "\{x_1/0\}"] \arrow[dr, Rightarrow, "{\{x_1/s(x_0),\, y_1/{:}(y_0, z)\}}", "\varepsilon"'] & \\ \mathit{nil} & {:}(y_0, \boxed{\mathit{take}(x_0, z)}) \end{tikzcd} \\
\begin{tikzcd} f(x_0) \mapsto f(x_0) \arrow[r, Rightarrow, "\varepsilon", "\varepsilon"'] & \boxed{\mathit{take}(x_0, \mathit{zeros})} \end{tikzcd}
\end{array}\right.$$

$$\mathcal{P}[\![\mathcal{R}]\!]{\uparrow}2 = \left\{\begin{array}{l}
\begin{tikzcd} \mathit{zeros} \mapsto \mathit{zeros} \arrow[r, Rightarrow, "\varepsilon", "\varepsilon"'] & {:}(0, \mathit{zeros}) \arrow[r, Rightarrow, "\varepsilon", "\varepsilon"'] & \boxed{{:}(0, {:}(0, \mathit{zeros}))} \end{tikzcd} \\
\begin{tikzcd} \mathit{take}(x_2, y_2) \mapsto \mathit{take}(x_2, y_2) \arrow[d, Rightarrow, "\varepsilon"', "\{x_2/0\}"] \arrow[dr, Rightarrow, "{\{x_2/s(x_1),\, y_2/{:}(y_1, z_1)\}}", "\varepsilon"'] & & \\ \mathit{nil} & {:}(y_1, \mathit{take}(x_1, z_1)) \arrow[dl, Rightarrow, "\varepsilon", "\varepsilon"'] \arrow[dr, Rightarrow, "{\{x_1/s(x_0),\, z_1/{:}(y_0, z_0)\}}", "\varepsilon"'] & \\ {:}(y_1, \mathit{nil}) & & {:}(y_1, {:}(y_0, \mathit{take}(x_0, z_0))) \end{tikzcd} \\
\begin{tikzcd} f(x_1) \mapsto f(x_1) \arrow[d, Rightarrow, "\varepsilon", "\varepsilon"'] & & \\ \mathit{take}(x_1, \mathit{zeros}) \arrow[d, Rightarrow, "\{x_1/0\}"', "\varepsilon"] \arrow[dr, Rightarrow, "\varepsilon", "\varepsilon"'] & & \\ \mathit{nil} & \mathit{take}(x_1, {:}(0, \mathit{zeros})) \arrow[d, Rightarrow, "\{x_1/0\}"', "\varepsilon"] \arrow[dr, Rightarrow, "\{x_1/s(x_0)\}", "\varepsilon"'] & \\ & \mathit{nil} & {:}(0, \mathit{take}(x_0, \mathit{zeros})) \end{tikzcd}
\end{array}\right.$$

Note that, to highlight the construction order of the various subtrees, we used indices in variables names that respect the order of introduction and boxed the subtrees which correspond to the previous iterate. By continuing the computation of the iterates, we obtain


$\mathcal{F}[\![\mathcal{R}]\!] =$
$zeros \mapsto zeros \xrightarrow[\varepsilon]{\varepsilon} :(0, zeros) \xrightarrow[\varepsilon]{\varepsilon} :(0, :(0, zeros)) \cdots$
$take(x,y) \mapsto take(x,y)$
$\{x/0\}$, ε
nil
$\{x/s(x'), y/:(y',z)\}$, ε
$:(y', take(x',z))$
$:(y', nil)$
$\{x'/s(x''), z/:(y'',z')\}$, ε
$:(y', :(y'', take(x'', z')))$
$f(x) \mapsto f(x)$
$take(x, zeros)$
nil
$take(x, :(0, zeros))$
$\{x/s(x')\}$
$:(0, take(x', zeros))$
$take(x, :(0, :(0, zeros)))$
$\{x'/0\}$
$:(0, nil)$
$\{x'/s(x'')\}$
$:(0, take(x'', :(0, zeros)))$

We can observe that, since terms in $\mathcal{R}$ are linear, the linearizing narrowing trees have just ε as delinearizers and actually they are isomorphic to full narrowing trees.

3.3.1 Properties of the TRS fixpoint denotation

The top-down goal-dependent denotation $\mathcal{O}$ and the bottom-up goal-independent denotation $\mathcal{F}$ are actually equivalent.

Theorem 3.30 (Equivalence of denotations) *Let* $\mathcal{R} \in \mathbb{R}_\Sigma$. *Then* $\mathcal{O}[\![\mathcal{R}]\!] = \mathcal{F}[\![\mathcal{R}]\!]$.

A straightforward consequence of Theorem 3.30 and Corollary 3.24 is

Corollary 3.31 (Correctness and full abstraction of $\mathcal{F}$) *Let* $\mathcal{R}_1, \mathcal{R}_2 \in \mathbb{R}_\Sigma$. *Then* $\mathcal{F}[\![\mathcal{R}_1]\!] = \mathcal{F}[\![\mathcal{R}_2]\!]$ *if and only if* $\mathcal{R}_1 \approx_{ss} \mathcal{R}_2$.

4 Conclusions

We have presented a condensed compositional bottom-up semantics for the *full class* of term rewriting systems which is fully abstract w.r.t. the *small-step* behavior of rewriting.

We are going to use this semantics to define, by abstraction, a condensed bottom-up semantics for the *full class* of term rewriting systems which is fully abstract w.r.t. the *big-step* behavior of rewriting and thus suitable for semantic-based program manipulation tools. Actually we have developed a first proposal of such a semantics (by following this approach) but restricted to the class of left-linear TRSs. We already used it to develop a semantics-based automatic specification synthesis prototype [6] which is giving promising results.

However, we believe that our notion of linearized narrowing which, differently from narrowing, represents faithfully the small-step behavior of rewriting (in a compact way), could be interesting for other applications as well.

References

[1] M. Alpuente, M. Comini, S. Escobar, M. Falaschi, and J. Iborra. A Compact Fixpoint Semantics for Term Rewriting Systems. *Theoretical Computer Science*, 411(37):3348–3371, 2010. 1, 3.27

[2] M. Alpuente, M. Comini, S. Escobar, M. Falaschi, and S. Lucas. Abstract Diagnosis of Functional Programs. In M. Leuschel, editor, *Logic Based Program Synthesis and Transformation – 12th International Workshop, LOPSTR 2002, Revised Selected Papers*, volume 2664 of *Lecture Notes in Computer Science*, pages 1–16, Berlin, 2003. Springer-Verlag. 1

[3] G. Bacci. *An Abstract Interpretation Framework for Semantics and Diagnosis of Lazy Functional-Logic Languages*. PhD thesis, Dipartimento di matematica e Informatica, Università di Udine, 2011. 1

[4] G. Bacci and M. Comini. A Fully-Abstract Condensed Goal-Independent Bottom-Up Fixpoint Modeling of the Behaviour of First Order Curry. Submitted for Publication., 2012. 1

[5] M. Comini, G. Levi, M. C. Meo, and G. Vitiello. Abstract Diagnosis. *Journal of Logic Programming*, 39(1-3):43–93, 1999. 1

[6] M. Comini and L. Torella. TRSynth: a Tool for Automatic Inference of Term Equivalence in Left-linear Term Rewriting Systems. In E. Albert and S.-C. Mu, editors, *PEPM '13, Proceedings of the ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation*, pages 67–70. Acm, 2013. 4

[7] J.-M. Hullot. Canonical Forms and Unification. In *Proceedings of the 5th International Conference on Automated Deduction*, volume 87 of *Lecture Notes in Computer Science*, pages 318–334, Berlin, 1980. Springer-Verlag. 1, 2, 3.1, 3.1.1

[8] A. Middeldorp and E. Hamoen. Completeness results for basic narrowing. *Applicable Algebra in Engineering, Communication and Computing*, 5:213–253, 1994. 10.1007/BF01190830. 3.1.1, A.1, A.2

[9] H. R. Nielson and F. Nielson. Infinitary Control Flow Analysis: a Collecting Semantics for Closure Analysis. In *Symposium on Principles of Programming Languages*, pages 332–345, 1997. 1

[10] TeReSe, editor. *Term Rewriting Systems*. Cambridge University Press, Cambridge, UK, 2003. 2

A Technical Proofs

Proof of Proposition 3.2.

Point 1 Implication $\Longleftarrow$ is straightforward. We prove $\Longrightarrow$ by reduction to absurd. Suppose that $\mathcal{R}_1 \approx_{ss} \mathcal{R}_2$ and $\mathcal{R}_1$ is not a variant of $\mathcal{R}_2$. Then there is at least one rule which is different in $\mathcal{R}_1$ w.r.t. $\mathcal{R}_2$ (or vice versa). Thus, by Equation (3.1), we can have rewriting steps which employ that rule in $\mathcal{B}^{ss}[\![\mathcal{R}_1]\!]$ which cannot be in $\mathcal{B}^{ss}[\![\mathcal{R}_2]\!]$ which is absurd.

Point 2 Similarly to previous point, but restricting to initial linear terms.

□

In the sequel we use the following results.

Proposition A.1 ([8]) *Suppose we have substitutions θ, ρ, ρ' and sets A, B of variables such that $(B - dom(\theta)) \cup range(\theta) \subseteq A$. If $\rho\restriction_A = \rho'\restriction_A$ then $(\theta\rho)\restriction_A = (\theta\rho')\restriction_A$.*

Proposition A.2 ([8]) *Let $\mathcal{R}$ be a TRS and suppose we have substitutions θ, ρ, ρ' and sets A, B of variables such that the following conditions are satisfied:*

$$\rho\restriction_A \ \mathcal{R}\text{-normalized} \qquad \theta\rho'\restriction_A = \rho\restriction_A \qquad B \subseteq (A - dom(\theta)) \cup range(\theta\restriction_A)$$

Then $\rho'\restriction_B$ is also $\mathcal{R}$-normalized.

Proof of Theorem 3.5. We prove the thesis by induction on the length of the rewriting derivation from t_0 to t_n. The case of length zero is straightforward.

Suppose $t_0 \to t_1 \to \ldots \to t_n$ is a rewriting derivation of length $n > 0$. We may assume, without loss of generality, that $var(l) \cap V = \varnothing$. We have $(s_0\eta_0)|_p = s_0|_p\eta_0 = \tau l$ for some substitution τ with $dom(\tau) \subseteq var(l)$. Let $\mu := \tau \cup \eta_0$. We have $s_0|_p\mu = s_0|_p\eta_0 = l\tau = l\mu$, so $s_0|_p$ and l are unifiable. Let $\theta'_1 := mgu(s_0|_p, l)$ and $\theta_1 := \theta'_1\restriction_{var(s_0|_p)}$. Clearly $dom(\theta_1) \cup range(\theta_1) \subseteq var(s_0|_p) \cup var(l)$. Moreover there exists a substitution ψ such that

$$\theta_1\psi = \mu \tag{1}$$

Now let $(s_1, \sigma_1) := lin(s_0[r\theta'_1]_p)$. By Definition 3.4

$$s_0 \xRightarrow[\sigma_1,\, l\to r]{\theta_1,\, p} s_1 \tag{2}$$

Let $V_1 := (V - dom(\theta_1)) \cup range(\theta_1) \cup dom(\sigma_1)$ and

$$\eta_1 := (\sigma_1\psi)\restriction_{V_1} \tag{3}$$

Clearly $dom(\eta_1) \subseteq V_1$. We have $var(s_1) = dom(\sigma_1) \cup var(s_0[r\theta_1]_p) \subseteq dom(\sigma_1) \cup var(s_0[l\theta_1]_p) = dom(\sigma_1) \cup var(s_0\theta_1) \subseteq V_1$. Therefore $var(s_1) \cup dom(\theta_1) \subseteq V_1$. By (1) and (3) we have $s_1\sigma_1\eta_1 = s_1\sigma_1\psi = s_0[r\theta_1]_p\psi = s_0[r]_p\theta_1\psi = s_0[r]_p\mu = s_0\mu[r\mu]_p$. Since $V \cap dom(\tau) = \varnothing$, we have

$$\mu\restriction_V = \eta_0\restriction_V. \tag{4}$$

Likewise $\mu\restriction_{var(r)} = \eta_0\restriction_{var(r)}$. Hence $s_0\mu[r\mu]_p = s_0\eta_0[r\tau]_p = t_0[r\tau]_p = t_1$. Thus

$$t_1 = s_1\sigma_1\eta_1. \tag{5}$$

Next we have to verify that $(\theta_1\eta_1)\restriction_V = \eta_0\restriction_V$. By (3) and by Proposition A.1 we have that $(\theta_1\eta_1)\restriction_V = (\theta_1\sigma_1\psi)\restriction_V$. By (1) and (4), $(\theta_1\eta_1)\restriction_V = (\theta_1\sigma_1\psi)\restriction_V$. Since $dom(\sigma_1) \notin V$, we have that $(\theta_1\sigma_1\psi)\restriction_V = (\theta_1\psi)\restriction_V = \mu\restriction_V = \eta_0\restriction_V$. Thus

$$(\theta_1\eta_1)\restriction_V = \eta_0\restriction_V. \tag{6}$$

Now we show that η_1 is $\mathcal{R}$-normalized. Since $dom(\eta_1) \subseteq V_1$, it suffices to show that $\eta_1\restriction_{V_1}$ is $\mathcal{R}$-normalized. Let $B := (V - dom(\theta_1)) \cup range(\theta_1\restriction_V)$. Proposition A.2 yields the normalization of $\eta_1\restriction_B$. Recall that $range(\theta_1) \subseteq var(s_0|_p) \cup var(l)$. Let $x \in range(\theta_1)$; since θ_1 is idempotent, clearly $x \notin dom(\theta_1)$. If $x \in var(s_0|_p) \subseteq V$ then $x \in V - dom(\theta_1) \subseteq B$. If $x \in var(l)$ then $x \in var(l\theta_1) = var(s_0|_p\theta_1)$ thus $x \in range(\theta_1\restriction_V) \subseteq B$. Thus $range(\theta_1) \subseteq B$ and then $V_1 = B \cup dom(\sigma_1)$. By this and (3), since $\eta_1\restriction_B$ is $\mathcal{R}$-normalized, $\eta_1\restriction_{V_1}$ is $\mathcal{R}$-normalized as well.

By inductive hypothesis we have a term s_n and substitutions θ', σ' and η_n such that

$$s_1 \overset{\theta'}{\underset{\sigma'}{\Longrightarrow}}{}^* s_n, \tag{7}$$

$$t_n = s_n\sigma'\eta_n, \tag{8}$$

$$(\theta'\eta_n)\restriction_{V_1} = \eta_1\restriction_{V_1}, \tag{9}$$

$$\eta_n \text{ is } \mathcal{R}\text{-normalized.} \tag{10}$$

Moreover, $s_1 \overset{\theta'}{\underset{\sigma',\mathcal{R}}{\Longrightarrow}}{}^* s_n$ and $t_1 \underset{\mathcal{R}}{\longrightarrow}{}^* t_n$ apply the same rewrite rules at the same positions. Let $\theta := \theta_1\theta'$ and $\sigma = \sigma_1\sigma'$. By (2) and (7) we obtain $s_0 \overset{\theta}{\underset{\sigma}{\Longrightarrow}}{}^* s_n$. By construction this narrowing derivation makes use of the same rewriting rules at the same positions as the rewriting derivation $t_0 \underset{\mathcal{R}}{\longrightarrow}{}^* t_n$. It remains to show that $(\theta\eta_n)\restriction_V = \eta_0\restriction_V$. By (9) and Proposition A.1, $(\theta_1\theta'\eta_n)\restriction_V = (\theta_1\eta_1)\restriction_V$. Therefore, by (6), $(\theta\eta_n)\restriction_V = (\theta_1\eta_1)\restriction_V = \eta_0\restriction_V$. □

Proof of Theorem 3.12. We prove $\Longrightarrow$ by reduction to absurd. Suppose that $\mathcal{R}_1 \approx_{ss} \mathcal{R}_2$ and $\exists t \in \mathcal{LT}(\Sigma,\mathcal{V})$. $\mathcal{N}[\![t \textit{ in } \mathcal{R}_1]\!] \neq \mathcal{N}[\![t \textit{ in } \mathcal{R}_2]\!]$. This means that there is at least one derivation which belongs to $\mathcal{N}[\![t \textit{ in } \mathcal{R}_1]\!]$ and not to $\mathcal{N}[\![t \textit{ in } \mathcal{R}_2]\!]$ (or vice versa). Hence, there is at least a rule which is not in common to the two programs. Thus $\mathcal{R}_1 \not\approx_{ss} \mathcal{R}_2$ which is absurd.

We prove $\Longleftarrow$ by reduction to absurd. Suppose that $\forall t \in \mathcal{LT}(\Sigma,\mathcal{V})$. $\mathcal{N}[\![t \textit{ in } \mathcal{R}_1]\!] = \mathcal{N}[\![t \textit{ in } \mathcal{R}_2]\!]$ and $\mathcal{R}_1 \not\approx_{ss} \mathcal{R}_2$. Then there is at least one rule which is different in $\mathcal{R}_1$ w.r.t. $\mathcal{R}_2$. Thus, by Equation (3.1) and Theorem 3.5, $\mathcal{N}[\![t \textit{ in } \mathcal{R}_1]\!] \neq \mathcal{N}[\![t \textit{ in } \mathcal{R}_2]\!]$ which is absurd. □

Proof of Theorem 3.14. We prove the two inclusions separately. Inclusion $\supseteq$ is immediate by Definition 3.13.

Inclusion $\subseteq$ is proved by reduction to absurd. Assume that there exists a derivation $t_0 \underset{\mathcal{R}}{\longrightarrow}{}^* t_n$ such that it does not belong to $\lfloor\mathcal{N}[\![t_0 \textit{ in } \mathcal{R}]\!]\rfloor$. Then, by Theorem 3.5, taken $\eta_0 = \varepsilon$, there exists a relaxed narrowing derivation of the form $s_0 \overset{\theta_1}{\underset{\sigma_1}{\Longrightarrow}} \ldots \overset{\theta_n}{\underset{\sigma_n}{\Longrightarrow}} s_n$. Note that, for each $i \in \{1..n\}$, θ_i and σ_i are just renaming, $\eta_i = \varepsilon$ and $t_i = s_i\sigma_i\ldots\sigma_0$. Let $\eta := \eta_0 = \varepsilon$. It is easy to see that there exists η' such that $\eta' = \theta_1 \uparrow \sigma_1 \uparrow \ldots \uparrow \theta_n \uparrow \sigma_n$. By Definition 3.13, there exists a rewriting derivation $t'_0 \underset{\mathcal{R}}{\longrightarrow}{}^* t'_n$ where $t'_i = s_i\eta'$ for each $i \in \{1..n\}$. Moreover, $dom(\sigma_k) \notin var(s_i)$ for each $k \in \{i+1..n\}$ and $\theta_i = (\sigma_j)\restriction_{dom(\theta_i)}$ or $\theta_i = (\sigma_j)\restriction_{dom(\theta_i)}$ for some j. Hence $s_i\eta' = s_i\sigma_i\ldots\sigma_0$. Thus $t'_i = t_i$ for each $i \in \{1..n\}$ which is absurd. □

Lemma A.3 *Let $\mathcal{R}$ a TRS, $x \in \mathcal{V}$, and $t, s \in \mathcal{T}(\Sigma,\mathcal{V})$ such that they do not share variables and $x \in t, x \notin s$. Then, $\mathcal{N}[\![t \textit{ in } \mathcal{R}]\!][x/\mathcal{N}[\![s \textit{ in } \mathcal{R}]\!]] = \mathcal{N}[\![t\{x/s\} \textit{ in } \mathcal{R}]\!]$*

Proof . It is sufficient to prove that, given $d_t \in \mathcal{N}[\![t \textit{ in } \mathcal{R}]\!]$ and $T_s = \mathcal{N}[\![s \textit{ in } \mathcal{R}]\!]$, if $d_t; x; T_s \vdash d$ then $d \in \mathcal{N}[\![t\{x/s\} \textit{ in } \mathcal{R}]\!]$. To this aim we will prove a stronger result: let $\pi \in \mathcal{T}(\mathcal{C},\mathcal{V})$, let $T_s = \mathcal{N}[\![s \textit{ in } \mathcal{R}]\!]$, and for any $d_t \in \mathcal{N}[\![t \textit{ in } \mathcal{R}]\!]$ such that $d_t; \pi; T_s \vdash d$ then $d \in \mathcal{N}[\![t\eta \textit{ in } \mathcal{R}]\!]$ where $\eta = \overleftarrow{mgu}(s,\pi)$. We proceed by structural induction on the proof tree.

rule 3.5a) straightforward.

rule 3.5b) By inductive hypothesis $d_t; \pi; T_{s'} \vdash d_{t\mu} \iff d_{t\mu} \in \mathcal{N}[\![t\mu \textit{ in } \mathcal{R}]\!]$. If it exists $d_{t\mu}$, then $t\mu|_p = s|_p$. Moreover, $\exists\mu'$ such that $\mu' = \overleftarrow{mgu}(s',\pi)$ and $t\mu'|_p = s'|_p$. Thus, $t\mu \overset{\theta,p}{\underset{\sigma}{\Longrightarrow}} d_{t\mu} \in \mathcal{N}[\![t\mu \textit{ in } \mathcal{R}]\!]$.

rule 3.5c) By inductive hypothesis we have that $d_{t_0};\pi_0;T_{s_0} \vdash d_{t_1} \iff d_{t_1} \in \mathcal{N}[\![t\mu_0 \mathrel{in} \mathcal{R}]\!], \dots, d_{t_n};\pi_n;T_{s_n} \vdash d_{t_{n+1}} \iff d_{t_{n+1}} \in \mathcal{N}[\![t\mu_n \mathrel{in} \mathcal{R}]\!]$, where $\mu_0 = \overleftarrow{mgu}(s_0,\pi_0), \dots, \mu_n = \overleftarrow{mgu}(s_n,\pi_n)$. We know that exists a linearizing narrowing step $t \overset{\theta,p}{\underset{\sigma}{\Longrightarrow}} t_0$ where $\theta_2 = mgu(t|_p, l), \theta = \theta_2\restriction_{var(t)}, (s,\sigma) = lin(r\theta_2), t\mu = t[s]_p$. We need to prove that it exists the linearizing narrowing step $t\mu \overset{\theta',p}{\underset{\sigma''}{\Longrightarrow}} t_{n+1}$. Since it exists $mgu(t|_p, l)$ and $mgu(t\mu, t\theta)$, then it exists θ_3 such that $\theta_3 = mgu((t\mu)|_p, l)$. Now let $\theta' := \theta_3\restriction_{var(t\mu)}$ and $(s',\sigma') = lin(r\theta_3)$, then $t\mu[s']_p = \bar{t}$. Let us observe that $t_1 = t_0\mu_0, t_2 = t_0\mu_0\mu_1, \dots, t_{n+1} = t_0\mu_0\dots\mu_n$ where $\mu_0 = \overleftarrow{mgu}(s_0,\pi_0), \dots, \mu_n = \overleftarrow{mgu}(s_n,\pi_n)$. Let $\mu^* = \mu_0\dots\mu_n$, then $t_{n+1} = t_0\mu^* = t\mu^*[s\mu^*]_p = t\mu[s\mu^*]_p = \bar{t}$. □

□

Proof of Theorem 3.23. We proceed by structural induction on term t

t* = *x Immediate by Equation (3.6a).

$t = \varphi(\overrightarrow{t_n})$

$$\begin{aligned}
&\mathcal{E}[\![\varphi(\overrightarrow{t_n})]\!]_{\mathcal{O}[\![\mathcal{R}]\!]} = \\
&\qquad [\text{by Equation (3.6b)}] \\
&= \mathcal{O}[\![\mathcal{R}]\!](\varphi(\overrightarrow{x_n}))[x_1/\mathcal{E}[\![t_1]\!]_{\mathcal{O}[\![\mathcal{R}]\!]}]\dots[x_n/\mathcal{E}[\![t_n]\!]_{\mathcal{O}[\![\mathcal{R}]\!]}] = \\
&\qquad [\text{by inductive hypothesis}] \\
&= \mathcal{N}[\![\varphi(\overrightarrow{x_n}) \mathrel{in} \mathcal{R}]\!][x_1/\mathcal{N}[\![t_1 \mathrel{in} \mathcal{R}]\!]]\dots[x_n/\mathcal{N}[\![t_n \mathrel{in} \mathcal{R}]\!]] = \\
&\qquad [\text{by Lemma A.3}] \\
&= \mathcal{N}[\![\varphi(\overrightarrow{t_n}) \mathrel{in} \mathcal{R}]\!]
\end{aligned}$$

□

Proof of Proposition 3.26. It is straightforward to prove that $\mathcal{P}[\![P]\!]$ is monotone and finitary, thus it is continuous. □

Proof of Theorem 3.30. We prove the two inclusions separately.

⊑) Let indicate with $\mathcal{O}_k$ all derivations of $\mathcal{O}[\![\mathcal{R}]\!]$ with length $\leq k$. by induction we prove that $\forall k\ \mathcal{O}_k \sqsubseteq \mathcal{P}[\![\mathcal{R}]\!]\uparrow k$.

k = 0) immediate.

k > 0) $\forall d \in \mathcal{O}_k$, we have that $d = f(x) \overset{\theta,\Lambda}{\underset{\sigma}{\Longrightarrow}} d'$ such that (by Theorem 3.23) $d'_t \in \mathcal{E}[\![t]\!]_{\mathcal{O}_{k-1}}$. Thus, by monotonicity of $\mathcal{E}$, we have that $d'_t \in \mathcal{E}[\![t]\!]_{\mathcal{P}[\![\mathcal{R}]\!]\uparrow k-1}$. By the definition of $\mathcal{P}$, $d \in \mathcal{P}[\![P]\!]_{\mathcal{P}[\![\mathcal{R}]\!]\uparrow k-1} = \mathcal{P}[\![\mathcal{R}]\!]\uparrow k$

Thus, by Proposition 3.26, $\mathcal{O}[\![\mathcal{R}]\!] = \bigsqcup_{k\geq 0}\mathcal{O}_k \sqsubseteq \bigsqcup_{k\geq 0}\mathcal{P}[\![\mathcal{R}]\!]\uparrow k = \mathcal{F}[\![\mathcal{R}]\!]$.

⊒) We need to prove that $f(\overrightarrow{x_n}) \overset{\theta,\Lambda}{\underset{\sigma}{\Longrightarrow}} \mathcal{E}[\![r']\!]_{\mathcal{O}[\![\mathcal{R}]\!]} \sqsubseteq \mathcal{N}[\![f(\overrightarrow{x_n}) \mathrel{in} \mathcal{R}]\!]$. By Theorem 3.23, $\mathcal{E}[\![r']\!]_{\mathcal{O}[\![\mathcal{R}]\!]}$ is a linearizing narrowing tree. Since $\theta = mgu(f(\overrightarrow{x_n}, l))\restriction_{\overrightarrow{x_n}}$ and $(\sigma, r') = lin(r)$, we can conclude that $f(\overrightarrow{x_n}) \overset{\theta,\Lambda}{\underset{\sigma}{\Longrightarrow}} \mathcal{E}[\![r']\!]_{\mathcal{O}[\![\mathcal{R}]\!]} \sqsubseteq \mathcal{N}[\![f(\overrightarrow{x_n}) \mathrel{in} \mathcal{R}]\!]$.

□