Term-graph rewriting via explicit paths

Emilie Balland, Pierre-Etienne Moreau

Term-graph rewriting via explicit paths

Emilie Balland and Pierre-Etienne Moreau

UHP & LORIA, and INRIA & LORIA,
BP 101, 54602 Villers-lès-Nancy Cedex France
{Emilie.Balland,Pierre-Etienne.Moreau}@loria.fr

Abstract. The notion of path is classical in graph theory but not directly used in the term rewriting community. The main idea of this work is to raise the notion of path to the level of first-order terms, i.e. paths become *part* of the terms and not just meta-information about them. These paths are represented by words of integers (positive or negative) and are interpreted as relative addresses in terms. In this way, paths can also be seen as a generalization of the classical notion of position for the first-order terms and are inspired by de Bruijn indexes.
In this paper, we define an original framework called Referenced Term Rewriting where paths are used to represent pointers between subterms. Using this approach, any term-graph rewriting systems can be simulated using a term rewrite-based environment.

1 Introduction

The notion of position is of course central as soon as one deals with data structures. Absolute as well as relative positions are at the heart of algorithms manipulating data structures and their appropriate use and representation can lead to very important differences in the complexity behavior and more generally in the expression of the algorithms themselves. A typical example is the notion of de Bruijn indexes for lambda-terms [11] that not only allows for an easier expression of data-structure manipulations, typically substitution, but also completely changes the way the algorithm is designed, because in this case alpha-conversion is useless.

The main idea of this paper is to make the notion of path first-class, i.e. paths become *part* of the terms and not just meta-information about them. Paths are defined as a generalization of positions and denote a relation from a source position to a target one. A main difference with classical positions that specify a subterm with respect to a global term is that the source position is not necessarily the root.

The first contribution of the paper is to introduce the notion of referenced terms to ground an extension of term rewriting where paths are used to express references and thus to provide a natural way to add pointers in classical terms. For instance, the term $f(a, g(a))$ where we want to make explicit the fact that the subterm a is shared, will be represented as the referenced term $f(a, g(\text{-}1.\text{-}2.1))$, where -1 and -2 denote backward move from respectively the first subterm of g

and the second subterm of f. The rational term $g(g(g(\ldots)))$ (a rational term is a possibly infinite term with finitely many subterms, see [10]) is represented by the referenced term $g(\text{-}1)$.

Based on the formalization of paths and a notion of rewrite relation for referenced terms, a strong contribution of this paper is to establish a simulation of term-graph rewriting by referenced term rewriting. Since this simulation is completely based on standard first-order terms, another main interest of the approach is to provide a safe and efficient way to represent and transform term-graphs in a purely term-rewriting based language. Beside the theoretical interest, this is very useful to implement program analysis tools where the representation, the analysis, and the transformation of control-flow and data-flow graphs are crucial. This new representation of terms generalizes standard first-order terms. It requires us to carefully design this new notion of terms and its use to get syntactic correctness. For instance, $g(\text{-}1\,.\,\text{-}1)$ makes no sense as such. Completeness with respect to the standard notions of term and term-graph rewritings have also to be established. This leads to an original and clean way for representing and transforming graphs in a maximally shared rewrite-based environment, making in particular possible the use of term rewriting strategies [16, 20] for term-graph rewriting.

The paper is organized as follows. In Section 2, we formalize the notion of paths and referenced terms where paths are interpreted as pointers. Section 3 shows the relation between referenced terms and cyclic term-graphs. In this section, paths are considered as a way to identify shared parts of the term. Sections 4 presents related work as well as the implementation that has been done in the Tom system. Section 5 concludes.

2 Paths and Referenced terms

We assume the reader to be familiar with the basic definitions of first-order terms given, in particular, in [4]. We briefly recall or introduce notations for a few concepts that will be used throughout this paper.

A signature $\mathcal{F}$ is a set of function symbols, each one associated to a natural number by an arity function. $\mathcal{T}(\mathcal{F}, \mathcal{X})$ is the set of *terms* built from a given finite set $\mathcal{F}$ of function symbols and a denumerable set $\mathcal{X}$ of variables. $\texttt{symb}(t)$ is a partial function from $\mathcal{T}(\mathcal{F}, \mathcal{X})$ to $\mathcal{F}$, which associates to each term t its top-symbol $f \in \mathcal{F}$. The set of variables occurring in a term t is denoted by $\mathcal{V}ar(t)$. If $\mathcal{V}ar(t)$ is empty, t is called a *ground term* and we note $\mathcal{T}(\mathcal{F}) = \mathcal{T}(\mathcal{F}, \emptyset)$ the set of ground terms. Given a set of terms $\mathcal{T}(\mathcal{F}, \mathcal{X})$, a *substitution* σ is a function from $\mathcal{X}$ to $\mathcal{T}(\mathcal{F})$, denoted $\sigma = \{x_1 \mapsto t_1, \ldots, x_k \mapsto t_k\}$ when its domain is finite. $\mathcal{R}an(\sigma)$ denotes its codomain. By abuse of notation, we mix up the term $a \in \mathcal{T}(\mathcal{F})$ and the function symbol $a \in \mathcal{F}$ when the arity of a is 0.

In term rewriting, the concept of positions is used to denote a subterm in a global term (i.e. the path from the root to this subterm). In this section, we define the notion of *path*, which generalizes the notion of position by denoting a

path from a subterm to another one, and not only the path from the root to a given subterm. Negative numbers designate bottom-up displacements

Definition 1 (Path). *We denote $\mathcal{P}$ the set of words on $\mathbb{Z} \setminus \{0\}$. We denote ϵ the empty word, $p_1 \,.\, p_2$ the concatenation of two words p_1 and p_2 and $|\mathsf{p}|$ the length of a word p. We denote $\mathcal{P}^*$ the set $\mathcal{P} \setminus \{\epsilon\}$.*

Example 1. ϵ, $1 \,.\, 3$, and $\text{-}2 \,.\, 1 \,.\, 3$ are paths, elements of $\mathcal{P}$.

The notion of path is oriented and corresponds to the route from a subterm to another one. For example, considering the term $f(a, b)$, the path -1.2 describes how to reach the subterm b starting from a. The negative integer -1 denotes a backward move from a to f, whereas 2 goes from f to b. The *inverse* of a path p is denoted by $\overline{p}$ and can be calculated using the equations $\overline{\epsilon} = \epsilon$ and $\overline{i \,.\, p} = \overline{p} \,.\, \text{-}i$. For example, $\overline{\text{-}1 \,.\, 2} = \text{-}2 \,.\, 1$.

Note that positions are a subset of paths (paths only composed of positive integers). In the rest of the paper, they will be denoted by Greek letters ω, δ. The empty word ϵ is also a position and is generally called the top-position because in term-rewriting, positions are only interpreted from the root of the term. Given two positions, we will denote by $\sqsubseteq$ the classical relation of prefixation between two words ($\omega_1 \sqsubseteq \omega_2$ if there exists a position p such that $\omega_1 \,.\, p = \omega_2$). The subterm of t at position ω is denoted $t_{|\omega}$. The replacement at position ω of the subterm $t_{|\omega}$ by t' is written $t[t']_\omega$. The set of positions of t is denoted $\mathcal{P}os(t)$. Given two positions ω_{src} and ω_{dest} (for *source* and *destination*), note that the path $\overline{\omega_{src}} \,.\, \omega_{dest}$ corresponds to a path connecting $t_{|\omega_{src}}$ to $t_{|\omega_{dest}}$. We denote $<_{\mathcal{P}}$ the lexicographic order on positions. For example $\epsilon <_{\mathcal{P}} 1$ and $1 \,.\, 2 <_{\mathcal{P}} 1 \,.\, 3$.

If we want to use these paths to define term-graphs, it is necessary to consider equivalence classes. Informally, two paths are equivalent if for every source position, their target positions are equal. For example, paths $1 \,.\, 2 \,.\, \text{-}2$ and 1 are equivalent.

In the following we define the notion of *canonical form* as the smallest path of this equivalence class. Moreover, when interpreting a negative integer as a backward move from the i-th child to the father, we must ensure that if the previous move in the word is positive, it leads to the same i-th child. For example, the path $1 \,.\, \text{-}2$ cannot be considered as valid because a move downward to its first child is followed by a move backward from its second child. These observations lead us to introduce the notion of well-formed paths, as well as a constant $\perp$ for representing ill-formed paths.

Definition 2 (Canonical path and path equivalence). *The canonical form of a path $p \in \mathcal{P}$, denoted $(\!|p|\!)$, is obtained by maximal application of the rule $i \,.\, \text{-}i \to \epsilon$ if $i \in \mathbb{Z}^*$. Two paths p_1 and p_2 are said equivalent if $(\!|p_1|\!) = (\!|p_2|\!)$.*

It is easy to show that the rule using to obtain canonical paths is confluent and terminating.

Definition 3 (Well-formed path). *We introduce a constant $\perp$ for denoting ill-formed paths. A path $p \in \mathcal{P}$ is well-formed if $(\!|p|\!) \not\to_R^* \perp$ with R is defined by the rule $p \,.\, i \,.\, \text{-}j \,.\, p' \to \perp$ if $i > 0$, $j > 0$ and $i \neq j$*

Example 2. $1\,\texttt{.}\,2\,\texttt{.}\,\text{-}2$ and $2\,\texttt{.}\,3\,\texttt{.}\,\text{-}3\,\texttt{.}\,\text{-}2\,\texttt{.}\,1$ are well-formed paths, but $1\,\texttt{.}\,\text{-}2$ is not.

Note that positions can be seen as a subset of well-formed paths because they correspond to paths only composed of positive integers. Note also that the inverse preserves the well-formedness. On the other hand, the concatenation of two well-formed paths does not always lead to a well-formed path: 1 is well-formed, -2 is well-formed, but $1\,\texttt{.}\,\text{-}2$ is not.

From these definitions, we show how the notion of paths can be used to extend an algebraic signature in order to represent referenced terms.

2.1 Referenced Terms

A referenced term is a term whose leaves may be a path, which denotes a reference to another subterm.

Definition 4 (Referenced terms). *Given a set of symbols $\mathcal{F}$ and a set of variables $\mathcal{X}$ such that $\mathcal{F}$, $\mathcal{X}$, $\mathcal{P}$ are disjoint, we denote by $\mathcal{T}_r(\mathcal{F},\mathcal{X})$ the set of* referenced terms *$\mathcal{T}(\mathcal{F}\cup\mathcal{P},\mathcal{X})$, where the elements of $\mathcal{P}$ are symbols of arity 0.*

Example 3. For $\mathcal{F}=\{f,g,a\}$, a, $g(\text{-}1)$, and $g(f(a,\text{-}2\,\texttt{.}\,1))$ are *referenced terms*, elements of $\mathcal{T}_r(\mathcal{F})$. For any $\mathcal{F}$ and $\mathcal{X}$, we have $\mathcal{T}(\mathcal{F},\mathcal{X})\subset\mathcal{T}_r(\mathcal{F},\mathcal{X})$.

Definition 5 (Dereferencing). *Given $t\in\mathcal{T}_r(\mathcal{F},\mathcal{X})$ and $\omega\in\mathcal{P}os(t)$:*

$$\mathtt{deref}(t,\omega)=\begin{cases}(\!|\omega\,\texttt{.}\,\mathtt{symb}(t_{|\omega})|\!) & \text{if } \mathtt{symb}(t_{|\omega})\in\mathcal{P}\\ \omega & \text{otherwise.}\end{cases}$$

We recall that $\mathtt{symb}(t)$ is a partial function from $\mathcal{T}(\mathcal{F},\mathcal{X})$ to $\mathcal{F}$, which associates to each term t its top-symbol $f\in\mathcal{F}$. The operation $\mathtt{deref}(t,\omega)$ returns the position pointed by $t_{|\omega}$ when $\mathtt{symb}(t_{|\omega})\in\mathcal{P}$, and ω otherwise. For example, $\mathtt{deref}(g(\text{-}1),1)=\epsilon$, but $\mathtt{deref}(g(a),1)=1$. Note that when $\omega\,\texttt{.}\,\mathtt{symb}(t_{|\omega})$ is ill-formed, the result of $\mathtt{deref}(t,\omega)$ is meaningless.

We now introduce a notion of *valid referenced terms*. The first condition ensures that the value returned by $\mathtt{deref}(t,\omega)$ is a position of $\mathcal{P}os(t)$, and thus is a well-formed path. For example, $\mathtt{deref}(g(-2),1)$ is not a valid term. The second condition forbids pointers of pointers like in $f(-1\,\texttt{.}\,2,-2\,\texttt{.}\,1)$. This last requirement is introduced only for simplicity but is not mandatory in term-graph simulation. Indeed, it could be interesting to consider such terms for modeling imperative languages for example.

Definition 6 (Valid referenced terms). *A term $t\in\mathcal{T}_r(\mathcal{F},\mathcal{X})$ is a* valid referenced term *if $\forall\omega\in\mathcal{P}os(t)$ such that $\mathtt{symb}(t_{|\omega})\in\mathcal{P}$ we have:*

- $\mathtt{deref}(t,\omega)\in\mathcal{P}os(t)$,
- $\mathtt{symb}(t_{|\mathtt{deref}(t,\omega)})\notin\mathcal{P}^*$.

We denote by $\mathcal{T}_{vr}(\mathcal{F},\mathcal{X})$ the set of valid referenced terms *and $\mathcal{T}_{vr}(\mathcal{F})$ the set of ground* valid referenced terms.

Empty paths (denoted ϵ) are allowed in valid referenced terms in order to deal with degenerated cycles that can appear when applying a collapsing rule, e.g. a rule of the form $I(x) \to x$. In the term-graph rewriting formalism, a fresh constant called *black hole* and denoted by $\bullet$ is generally introduced [3]. In our context, it is not necessary since ϵ corresponds intuitively to this constant.

Example 4. The terms ϵ, $g(\text{-}1)$, $f(\text{-}1\,.\,2, a)$ and $f(\text{-}1\,.\,2, \epsilon)$ are valid, but $g(3)$ and $f(\text{-}1\,.\,2, \text{-}2)$ are not. Terms corresponding to non-empty paths (1, -1 . 2, etc.) are elements of $\mathcal{T}_r(\mathcal{F}, \mathcal{X})$ but are not valid (i.e. $\notin \mathcal{T}_{vr}(\mathcal{F}, \mathcal{X})$). The term $t = f(\text{-}1\,.\,1\,.\,\text{-}1, \text{-}2\,.\,3)$ is invalid because $\texttt{deref}(t, 2) = (\!|2\,.\,\text{-}2\,.\,3|\!) = 3$ which is not in $\mathcal{P}os(t) = \{\epsilon, 1, 2\}$.

3 Term-Graph Rewriting

There exist different formalisations of term-graph rewriting, category-theory oriented [15, 18], equationally oriented [3, 19] or implementation oriented [6]. The difference between terms and term-graphs is the notion of horizontal and vertical sharing. In this section, we base our work on the equational framework introduced in [3]. This well established framework allows the definition of possibly cyclic term-graphs, using systems of recursion equations.

Definition 7 (System of recursion equations from [3]). *Given a finite set $\mathcal{F}$ of function symbols and a denumerable set $\mathcal{X}$ of variables, a system of recursion equations is of the form $\{\alpha_1 \mid \alpha_1 = t_1, \ldots, \alpha_n = t_n\}$, $\forall i, j \in [1, n]$ $\alpha_i \in \mathcal{X}$, $\alpha_i \neq \alpha_j$, $t_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is of the form $f(\beta_1, \ldots, \beta_m)$, $f \in \mathcal{F}$, $\forall j \in [1, m]$ $\beta_j \in \mathcal{X}$. Moreover, $\forall i \in [1, n]$ α_i must be reachable from α_1.*

Given a system of recursion equations L, the root is denoted $\texttt{root}(L)$ and corresponds to the recursion variable α_1. The set of equations is denoted $\texttt{set}(L)$. A variable α is said *bound* when it appears in the left-hand side of an equation. Otherwise, α is *free*. Note that systems of recursion equations are considered modulo renaming of the recursion variables. In Definition 7, the systems of recursion equations have been presented in *flattened form* (t_i of the form $f(\beta_1, \ldots, \beta_m)$) which ensures the unicity of the representation of a term-graph (modulo renaming of recursion variables). An example of term-graph is given in Figure 1.

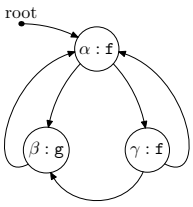
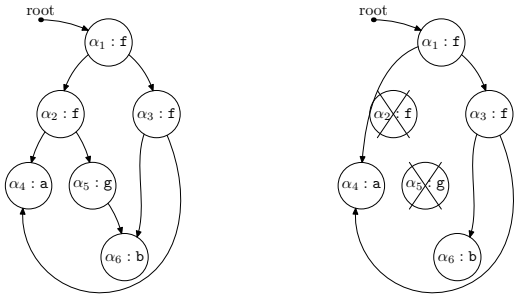


Fig. 1. This cyclic term-graph corresponds to the system of recursion equations $\{\alpha \mid \alpha = f(\beta, \gamma), \beta = g(\alpha), \gamma = f(\beta, \alpha)\}$. It contains horizontal sharing: α and γ share the same subterm β; as well as vertical sharing: α is a subterm of both β and γ.

Definition 8 (Equational term-graph rewriting [3]). *Given a rewrite rule composed of two systems of recursion equations L_1 and L_2 with the same root*

(L_1 and L_2 are not necessarily in flattened form) and where the free variables of L_2 are included in the set of free variables of L_1, a system of recursion equations L is rewritten into L' by the rule $L_1 \to L_2$ if there exists a variable substitution σ (Definition 4.1 of [3]) and a recursion equation $\alpha = t$ in L such that $\texttt{set}(\sigma(L_1)) \subseteq \texttt{set}(L)$ and $\alpha = \texttt{root}(\sigma(L_1))$. $\texttt{root}(L') = \texttt{root}(L)$ and $\texttt{set}(L') = \texttt{set}(L) \setminus \{\alpha = t\} \cup \texttt{set}(\sigma'(L_2))$ where $\sigma'(L_2)$ denotes $\sigma(L_2)$ in which every bound variable (except the root) has been renamed using a fresh name. To obtain from L' a system as defined in Definition 7, equations corresponding to unreachable bound recursion variables are removed and equations are flattened (see [3] for more details). Degenerated cycles, i.e. equations of the form $\alpha = \alpha$ are replaced by $\alpha = \bullet$. In case of equations of type $\alpha = \beta$, each occurrence of α is substituted by β and the equation $\alpha = \beta$ is removed.

Example 5. Suppose we want to apply the rule $\{\beta_1 \mid \beta_1 = f(\beta_2, \beta_3)\} \to \{\beta_1 \mid \beta_1 = \beta_2\}$, which corresponds to $f(x, y) \to x$, on the following term-graph:



initial subject s final result s'

The initial term-graph is $L = \{\alpha_1 \mid \alpha_1 = f(\alpha_2, \alpha_3), \alpha_2 = f(\alpha_4, \alpha_5), \alpha_3 = f(\alpha_6, \alpha_4), \alpha_4 = a, \alpha_5 = g(\alpha_6), \alpha_6 = b\}$. When applying the rule at position 1 (i.e. on α_2), we have $\sigma = \{\beta_1 \mapsto \alpha_2, \beta_2 \mapsto \alpha_4, \beta_3 \mapsto \alpha_5\}$ and α_2 is the selected bound variable. $\sigma(L_2) = \{\alpha_2 = \alpha_4\}$ (note that renaming with fresh variables is not necessary in this case) and we get $L' = L \setminus \{\alpha_2 = f(\alpha_4, \alpha_5)\} \cup \sigma(L_2)$ as final result. This corresponds to the system $\{\alpha_1 \mid \alpha_1 = f(\alpha_4, \alpha_3), \alpha_3 = f(\alpha_6, \alpha_4), \alpha_4 = a, \alpha_6 = b\}$ after cleanup.

3.1 Referenced term equivalence

In order to simulate term-graphs with referenced terms, we need to establish equivalence classes between valid referenced terms. For example, $f(\text{-}1\texttt{.}2, a)$ and $f(a, \text{-}2\texttt{.}1)$ should be equivalent. They both correspond to the term-graph rooted by f whose two children correspond to the shared subterm a. To define the equivalence, we introduce three intermediate functions that characterize translation, expansion and sharing.

The first one, called *subterm translation*, is essential in the following. Given a term t and two positions ω_1, ω_2, as illustrated in Figure 2, the translation in t from ω_1 to ω_2, denoted $t_{[\omega_1 \to \omega_2]}$, returns the subterm $t_{|\omega_1}$ where the references

contained in $t_{|\omega_1}$ that point outside $t_{|\omega_1}$ have been updated as if $t_{|\omega_1}$ was moved to the position ω_2.

Definition 9 (Subterm translation). *Given $t \in \mathcal{T}_{vr}(\mathcal{F}, \mathcal{X})$ and two positions ω_1, ω_2, we consider the* subterm translation $t_{[\omega_1 \twoheadrightarrow \omega_2]}$ *defined such that $\mathcal{P}os(t_{[\omega_1 \twoheadrightarrow \omega_2]}) = \mathcal{P}os(t_{|\omega_1})$ and $\forall \delta \in \mathcal{P}os(t_{|\omega_1})$:*

$$\mathit{symb}(t_{[\omega_1 \twoheadrightarrow \omega_2]_{|\delta}}) = \begin{cases} (\!|\overline{\omega_2 \,.\, \delta} \,.\, \mathit{deref}(t, \omega_1 \,.\, \delta)|\!) & \text{if } \mathit{symb}(t_{|\omega_1.\delta}) \in \mathcal{P}^* \\ & \text{and } \omega_1 \not\sqsubseteq \mathit{deref}(t, \omega_1 \,.\, \delta) \\ \mathit{symb}(t_{|\omega_1.\delta}) & \text{otherwise.} \end{cases}$$

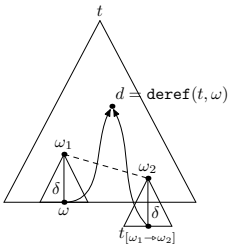


Fig. 2. Given t and ω_1, let us suppose that at position ω the subterm $t_{|\omega}$ contains a reference to the subterm d (i.e. $d = \mathtt{deref}(t, \omega)$). The translation from ω_1 to ω_2 corresponds to an update of $t_{|\omega_1}$ as if it was moved to ω_2. To maintain the pointers to the referenced terms, the paths stored in $t_{|\omega_1}$ are updated. The result of this operation is the updated subterm $t_{|\omega_1}$. For example, $f(g(\text{-}1 \,.\, \text{-}1), a)_{[1 \twoheadrightarrow 2]} = g(\text{-}1 \,.\, \text{-}2)$.

The operation *expansion* noted $\mathtt{exp}$ consists in replacing all the sharing by duplication. Given a set of function symbols $\mathcal{F}$, $\mathtt{exp}$ is a function from $\mathcal{T}_{vr}(\mathcal{F})$ to $\mathcal{T}^{\infty}(\mathcal{F} \cup \{\epsilon\})$ where $\mathcal{T}^{\infty}(\mathcal{F} \cup \{\epsilon\})$ is the set of infinite terms over $\mathcal{F} \cup \{\epsilon\}$ defined as partial functions from the infinite set of positions to $\mathcal{F} \cup \{\epsilon\}$. We denote $\bot$ the undefined term represented by the empty function $\emptyset \to \mathcal{F} \cup \{\epsilon\}$. See [9] for more details.

Definition 10 (Expansion). *Given $t \in \mathcal{T}_{vr}(\mathcal{F})$, we consider the chain $\{t_i\}_{i \in \mathbb{N}}$ of terms of $\mathcal{T}^{\infty}(\mathcal{F} \cup \mathcal{P})$ defined as follows:*

- $t_0 = t$
- $t_{n+1} = t_n[t_{n[\mathit{deref}(t_n, \omega) \twoheadrightarrow \omega]}]_{\omega}$
  where ω is the smallest position of $\mathcal{P}os(t_n)$ such that $\mathit{symb}(t_{n|\omega}) \in \mathcal{P}^*$
  (We consider the following order: $\omega < \omega'$ if $|\omega| < |\omega'| \ \vee \ (|\omega| = |\omega'| \wedge \omega <_{\mathcal{P}} \omega')$)

$\mathit{exp}(t) \in \mathcal{T}^{\infty}(\mathcal{F})$ is defined as $\bigcup_{i=0}^{\infty} t'_i$ where t'_i corresponds to t_i where every path $p \in \mathcal{P}^*$ has been replaced by $\bot$.

Proposition 1. *Given $t \in \mathcal{T}_{vr}(\mathcal{F})$, $\mathit{exp}(t)$ is a total function (i.e. total w.r.t. to the arities, not totally defined over the set of all positions. For more explanations, see [9]).*

Proof. By definition of the chain $\{t_i\}_{i \in \mathbb{N}}$, $\mathtt{exp}(t)$ is a function. Moreover, as every path is replaced by a subterm, $\mathtt{exp}(t)$ does not contain $\bot$ and the symbol arities are respected. □

Example 6. The function `exp` replaces in a valid referenced term every reference by the corresponding expanded subterm. $\texttt{exp}(f(\text{-}1\,.\,2, a)) = \texttt{exp}(f(a, \text{-}2\,.\,1)) = f(a, a)$ and $\texttt{exp}(g(\text{-}1)) = g(g(g(\ldots)))$. Note that in case of a cycle, the expanded term corresponds to an infinite term. A non-trivial example is $f(g(\text{-}1.\text{-}1.2), h(\text{-}1.\text{-}2\,.\,1))$. In this case, we need to update paths at every application of the rule. As the shared subterms are dependent, the result is infinite. We finally obtain $f(g(h(g(h(\ldots)))), h(g(h(g(h(\ldots))))))$.

Thus two equivalent referenced terms have the same expansion. However, this condition is necessary but not sufficient because the two terms to compare must also have a similar sharing. For example, $f(a, a)$ is not equivalent to $f(a, \text{-}2\,.\,1)$ because a is not explicitly shared in the first one. For this, we introduce a third relation called *sharing* which computes the set of shared positions.

Definition 11 (Sharing). *Given $t \in \mathcal{T}_{vr}(\mathcal{F})$, we consider:*

- $\mathit{share}_0(t) = \{\{\omega, \mathit{deref}(t, \omega)\} \mid \omega \in \mathcal{P}os(t)$ *and* $\mathit{symb}(t_{|\omega}) \in \mathcal{P}^*\}$
- $\mathit{share}_{n+1}(t) = \{\{\omega'\,.\,q, q'\} \mid \{\omega, \omega'\}, \{\omega\,.\,q, q'\} \in \mathit{share}_n(t)\}$

The function $\mathit{share}(t)$ *is defined as* $\bigcup_{n=0}^{\infty} \mathit{share}_n(t)$.

Example 7. The function `share` computes the set of shared position pairs. For example, $\texttt{share}(f(\text{-}1\,.\,2, a)) = \{\{1, 2\}\}$ and $\texttt{share}(g(\text{-}1)) = \{\{1, \epsilon\}\}$. A non-trivial example is $f(g(\text{-}1\,.\,\text{-}1\,.\,2), h(\text{-}1\,.\,\text{-}2\,.\,1))$. At the first step, $\texttt{share}_0(t) = \{\{1, 2\,.\,1\}, \{2, 1\,.\,1\}\}$. In this case, it is necessary to close the relation of sharing with prefixes. We finally obtain the infinite set of related positions $\texttt{share}(t) = \{\{1, 2.1\}, \{2, 1.1\}, \ldots, \{1.(1.1)^*, 2.1.(1.1)^*\}, \{2.(1.1)^*, 1.1.(1.1)^*\}\}$ (* denotes the sublist repetition). This infinite result is due to the inter-dependency of the two references.

Definition 12 (Equivalence). *Two valid referenced terms t_1, t_2 are equivalent (denoted by $t_1 \backsim t_2$) if* $\mathit{share}(t_1) = \mathit{share}(t_2)$ *and* $\mathit{exp}(t_1) = \mathit{exp}(t_2)$.

It is easy to show that $\backsim$ is an equivalence relation.

3.2 Canonical referenced terms

For every equivalence class, we define a canonical form using $<_{\mathcal{P}}$, the lexicographic order on positions.

Definition 13 (Canonical referenced terms). *A valid referenced term $t \in \mathcal{T}_{vr}(\mathcal{F}, \mathcal{X})$ is canonical if for every position $\omega \in \mathcal{P}os(t)$ such that* $\mathit{symb}(t_{|\omega}) \in \mathcal{P}^*$, $\mathit{symb}(t_{|\omega})$ *is a canonical path and* $\mathit{deref}(t, \omega) <_{\mathcal{P}} \omega$.
We denote by $\mathcal{T}_g(\mathcal{F}, \mathcal{X})$ the set of canonical referenced terms and $\mathcal{T}_g(\mathcal{F})$ the set of ground canonical referenced terms.

Example 8. The term $f(a, \text{-}2 \,.\, 1)$ is canonical but $f(\text{-}1 \,.\, 2, a)$ is not because $\texttt{deref}(f(\text{-}1 \,.\, 2, a), 1) = 2$ (the position of the pointed subterm a) is not smaller than 1.

To define the normalization function that returns the canonical form of any valid referenced term, we introduce a swapping function that permutes two subterms and updates all the paths contained in the global term in order to preserve the sharing. First, we translate the two subterms. This translation updates the pointers from the subterms to the external context. To obtain a valid referenced term, we still have to update every pointer from the outside to the subterms.

Definition 14 (Swapping). *Given $t \in \mathcal{T}_{vr}(\mathcal{F}, \mathcal{X})$ and two disjoint positions ω_1, ω_2 ($\omega_1 \not\sqsubseteq \omega_2$ and $\omega_2 \not\sqsubseteq \omega_1$), we consider $u = t_{[\omega_1 \twoheadrightarrow \omega_2]}$, $v = t_{[\omega_2 \twoheadrightarrow \omega_1]}$, $t' = t[v]_{\omega_1}[u]_{\omega_2}$. The swapping in t of the subterms at position ω_1 and ω_2 is denoted by $t_{[\omega_1 \leftrightarrow \omega_2]}$, and is defined such that $\forall \omega \in \mathcal{P}os(t')$, we have:*

$$\mathit{symb}(t_{[\omega_1 \leftrightarrow \omega_2]}{}_{|\omega}) = \begin{cases} (\!|\overline{\omega} \,.\, \omega_2 \,.\, \delta|\!) \text{ if } \mathit{symb}(t'_{|\omega}) \in \mathcal{P}^*,\ \omega \not\sqsubseteq \omega_1 \\ \qquad\qquad \text{and } \exists \delta \text{ s.t. } \mathit{deref}(t', \omega) = \omega_1 \,.\, \delta \\ (\!|\overline{\omega} \,.\, \omega_1 \,.\, \delta|\!) \text{ if } \mathit{symb}(t'_{|\omega}) \in \mathcal{P}^*,\ \omega \not\sqsubseteq \omega_2 \\ \qquad\qquad \text{and } \exists \delta \text{ s.t. } \mathit{deref}(t', \omega) = \omega_2 \,.\, \delta \\ \mathit{symb}(t'_{|\omega}) \text{ otherwise.} \end{cases}$$

Example 9. $f(a,b)_{[1 \leftrightarrow 2]} = f(b,a)$, $f(g(\text{-}1 \,.\, \text{-}1 \,.\, 2), h(\text{-}1 \,.\, \text{-}2 \,.\, 1))_{[1 \leftrightarrow 2]} = f(h(\text{-}1 \,.\, \text{-}1 \,.\, 2), g(\text{-}1 \,.\, \text{-}2 \,.\, 1))$. A more complex example is the swap of a and b in $t = f(f(a,b), \text{-}2 \,.\, 1 \,.\, 2)$. In this case, the reference $\text{-}2 \,.\, 1 \,.\, 2$ has to be updated because it has to reference b. Thus, the result is $f(f(b,a), \text{-}2 \,.\, 1 \,.\, 1)$.

The swapping function preserves the notion of validity (Definition 6). Its complexity is linear on the size of the term since in the worst case, all the references in the term must be updated. In the following, we introduce a *normalization function* that associates to every valid referenced term its canonical form.

Definition 15 (Normalization). *We define $[\![\ ]\!] : \mathcal{T}_{vr}(\mathcal{F}, \mathcal{X}) \to \mathcal{T}_g(\mathcal{F}, \mathcal{X})$ such that given $t \in \mathcal{T}_{vr}(\mathcal{F}, \mathcal{X})$, $[\![t]\!]$ is the normal form of t' (t where every path is in canonical form) with respect to the conditional rule: $t' \to t'_{[\omega \leftrightarrow \mathit{deref}(t', \omega)]}$ if $\omega <_{\mathcal{P}} \mathit{deref}(t', \omega)$. The proof of this rule convergence can be found in the Appendix A.*

Example 10. $[\![a]\!] = a$, $[\![f(\text{-}1 \,.\, 2, a)]\!] = f(a, \text{-}2 \,.\, 1)$, and $[\![f(g(\text{-}1 \,.\, \text{-}1 \,.\, 2), h(\text{-}1 \,.\, \text{-}2 \,.\, 1))]\!] = f(g(h(\text{-}1 \,.\, \text{-}1)), \text{-}2 \,.\, 1 \,.\, 1)$

Note that the normalization is linear in the size of the term when the swapping is applied in a leftmost-innermost way.

Proposition 2. *$\forall t \in \mathcal{T}_{vr}(\mathcal{F}, \mathcal{X})$, we have $t \backsim [\![t]\!]$.*

Proof. Given $t \in \mathcal{T}_{vr}(\mathcal{F}, \mathcal{X})$, t' is trivially equivalent to t and every rewriting step of normalization preserves the two functions $\texttt{share}(t)$ and $\texttt{exp}(t)$. In fact, the swapping between a pointer and its corresponding pointed subterm preserves the sharing and as only references are updated, the expansion is the same. □

Proposition 3. *$\forall t_1, t_2 \in \mathcal{T}_{vr}(\mathcal{F}, \mathcal{X})$, we have: $[\![t_1]\!] \backsim [\![t_2]\!] \Leftrightarrow [\![t_1]\!] = [\![t_2]\!]$.*

Proof. First, the proof that $[\![t_1]\!] = [\![t_2]\!] \Rightarrow [\![t_1]\!] \backsim [\![t_2]\!]$ is trivial because as $\backsim$ is an equivalence relation and thus is reflexive. Secondly, we prove that $[\![t_1]\!] \backsim [\![t_2]\!] \Rightarrow [\![t_1]\!] = [\![t_2]\!]$. Suppose they are not equal, it means that there exists a shared subterm at a position ω referenced at a position ω' in $[\![t_1]\!]$ and the contrary in $[\![t_2]\!]$. As $[\![t_1]\!], [\![t_2]\!]$ are canonical, it implies that $\omega <_{\mathcal{P}} \omega'$ and $\omega' <_{\mathcal{P}} \omega$ which is impossible due to the total order on positions. So $[\![t_1]\!] = [\![t_2]\!]$. □

Theorem 1. *$\forall t_1, t_2 \in \mathcal{T}_{vr}(\mathcal{F}, \mathcal{X})$, we have: $t_1 \backsim t_2 \Leftrightarrow [\![t_1]\!] = [\![t_2]\!]$*

Proof. Direct consequence of the propositions 2 and 3. □

In practice, to verify that two terms are equivalent we compare their canonical forms. It is simpler and more realistic than computing $\texttt{exp}(t)$ and $\texttt{share}(t)$, which can be infinite.

3.3 Term-graph rewriting using canonical referenced terms

We introduce an original algorithm for implementing term-graph rewriting using canonical referenced terms. By manipulating only canonical referenced terms, we obtain a mapping one-to-one with term-graphs which makes easier the encoding of matching and rewriting. In Figure 3 we present a set of rules ($\mathcal{T}_g$-Matching), which is a specialization of the syntactic pattern matching algorithm presented in [17].

Decompose	$E \wedge f(p_1, \ldots, p_n) \ll^{s,\omega}_{\delta} f(t_1, \ldots, t_n) \parallel \Delta$	$\mapsto\!\!\!\!\mapsto E \wedge \bigwedge_{i=1}^{n} p_i \ll^{s,\omega\cdot i}_{\delta\cdot i} t_i \parallel \Delta \cup \{(\delta, \omega)\}$
Variable	$E \wedge x \ll^{s,\omega}_{\delta} t \parallel \Delta$	$\mapsto\!\!\!\!\mapsto E \parallel \Delta \cup \{(\delta, \omega)\}$
Stability	$E \wedge \pi \ll^{s,\omega}_{\delta} f(t_1, \ldots, t_n) \parallel \Delta$	$\mapsto\!\!\!\!\mapsto E \parallel \Delta \cup \{(\delta, \omega)\}$ if $(\!(\delta \cdot \texttt{symb}(\pi))\!), \omega) \in \Delta$
Dereferencing	$E \wedge p \ll^{s,\omega}_{\delta} \pi \parallel \Delta$	$\mapsto\!\!\!\!\mapsto E \wedge p \ll^{s,\omega'}_{\delta} s_{\lvert\omega'} \parallel \Delta \cup \{(\delta, \omega)\}$ where $\omega' = (\!(\omega \cdot \texttt{symb}(\pi))\!)$

Fig. 3. We consider the set of rules $\mathcal{T}_g$-Matching where E is a conjunction of constraints, Δ is a set of pairs, x is a variable ($\in \mathcal{X}$), f is a symbol, element of $\in \mathcal{F} \cup \{\epsilon\}$, (remember that ϵ corresponds to $\bullet$), s, t, $t_1, \ldots, t_n$ are ground referenced term ($\in \mathcal{T}_g(\mathcal{F})$), π is a non-empty path ($\texttt{symb}(\pi) \in \mathcal{P}^*$), p is a pattern not reduced to a variable ($\in \mathcal{T}_g(\mathcal{F}, \mathcal{X}) \setminus \mathcal{X}$), p_i are patterns ($\in \mathcal{T}_g(\mathcal{F}, \mathcal{X})$), $\wedge$ is the classical boolean connector, which is associative and commutative. Starting from a constraint $l \ll^{s,\omega}_{\epsilon} s_{\lvert\omega} \parallel \emptyset$, the reduction leads either to $\top$ (the neutral element of $\wedge$) or a conjunction of matching constraints of the form $p \ll^{s,\omega}_{\delta} t$, where δ and ω correspond to the positions of p and t with respect to l and s (i.e. $p = l_{\lvert\delta}$ and $t = s_{\lvert\omega}$). The context Δ corresponds to the set of positions already visited. This set is necessary to correctly handle the case of cyclic terms.

Definition 16 (Rule application). *Given $s \in \mathcal{T}_g(\mathcal{F})$ and $l, r \in \mathcal{T}_g(\mathcal{F}, \mathcal{X})$, the rule $l \to r$ can be* applied *to subject s at position ω if $l \ll_\epsilon^{s,\omega} s_{|\omega} \parallel \emptyset$ reduces to $\top \parallel \Delta$ by application of $\mathcal{T}_g$-Matching.*

Note that the algorithm given Figure 3 does not compute a substitution. Moreover, contrary to syntactic term matching algorithms, there is no rule for handling variables that have multiple occurrences. The notion of non-linearity in *term rewriting* should not be confused with the notion of non-linearity in *term-graph rewriting*. The latter one corresponds to subterms sharing. For example, $\{\alpha \mid \alpha = f(\beta, \beta)\}$ denotes a term-graph where the two subterms of f are *shared*. This does not match $\{\alpha \mid \alpha = f(\beta, \gamma), \beta = a, \gamma = a\}$. In our formalism, *linear* term-rewriting is sufficient to simulate *non-linear* term-graph rewriting. For example, the system of recursion equations $\{\alpha \mid \alpha = f(\beta, \beta)\}$ can be encoded by $f(x, -2 \centerdot 1)$, where x appears only once.

Proposition 4. *Given a subject $s \in \mathcal{T}_g(\mathcal{F})$, a pattern $l \in \mathcal{T}_g(\mathcal{F}, \mathcal{X})$, a position $\omega \in \mathcal{P}os(s)$, the reduction of $l \ll_\epsilon^{s,\omega} s_{|\omega} \parallel \emptyset$ by $\mathcal{T}_g$-Matching is convergent.*

Proof. First, we prove the termination. We consider the lexicographic combination of prefix ordering on positions ($\sqsubset$) and $<_{\mathcal{P}}$. This strict order is well-founded because $\mathcal{P}os(l) \times \mathcal{P}os(s)$ is finite. Its multiset extension to $\biguplus_{(p \ll_\delta^{s,\omega} t) \in E}(\delta, \omega)$ decreases at each application of $\mathcal{T}_g$-Matching rules.

Secondly, proving the local confluence is trivial because there is no interference between the rules. When two rules r_1 and r_2 can be applied on a subject t, we obtain the same result t' when applying r_1 followed by r_2 or r_2 followed by r_1. As $\mathcal{T}_g$-Matching terminates, local confluence implies convergence. □

Definition 17 (Rewriting algorithm). *Given $t \in \mathcal{T}_g(\mathcal{F})$ and $l, r \in \mathcal{T}_g(\mathcal{F}, \mathcal{X})$:*

- *l and r are both linear.*
- *we denote by ω_{xl} the position of the variable x in l.*
- *t is rewritten into t' by the rule $l \to r$ if:*
  1. *there exists a position ω such that the rule can be applied to t (following the Definition 16),*
  2. *$t' = [\![\langle \dot{t}, \dot{r} \rangle_{[1 \centerdot \omega \leftrightarrow 2]}]\!]_{|1}$, where*
     $\langle \_, \_ \rangle$ is a fresh binary symbol,
     $\dot{t}$ corresponds to t where every path towards the position $1 \centerdot \omega$ has been replaced by a path towards the position 2,
     $\dot{r}$ is the ground term corresponding to r in which the occurrence of a variable x (whose position is denoted by ω_{xr}) is replaced by the path $(\!|\overline{\omega_{xr}} \centerdot \text{-}2 \centerdot \mathit{deref}(\langle \dot{t}, r \rangle, 1 \centerdot \omega \centerdot \omega_{xl})|\!)$.

In this algorithm, no substitution is explicitly computed. Instead, the right-hand side of the rule is instantiated by replacing every variable by paths to their corresponding subterm in $t_{|\omega}$. The binary symbol $\langle \_, \_ \rangle$ enables to connect it with the global subject in a valid referenced term $\langle \dot{t}, \dot{r} \rangle$. By swapping the redex and the right-hand side of the rule in $\langle \dot{t}, \dot{r} \rangle_{[1 \centerdot \omega \leftrightarrow 2]}$, the substitution is automatically

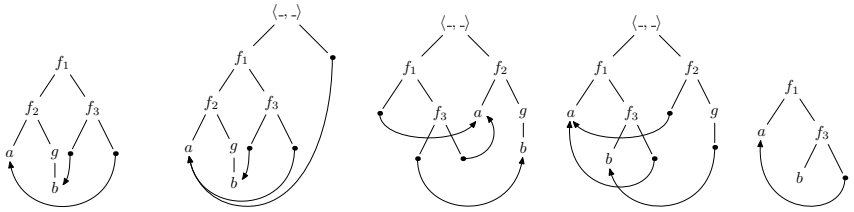
applied. The main subtlety of the algorithm is to rebalance the whole term using normalization. All the shared subterms in $t_{|\omega}$ that must be conserved are moved to the term and the unused part is left in the right part of $\langle \dot{t}, \dot{r} \rangle_{[1 \bullet \omega \leftrightarrow 2]}$. At the end, the result of the rewrite step corresponds exactly to the left child of $[\![\langle \dot{t}, \dot{r} \rangle_{[1 \bullet \omega \leftrightarrow 2]}]\!]$.

The complexity of the rewriting step is linear in the size of the global subject because the complexity of the swapping and the normalization are linear.

Example 11. Suppose we want to apply the rule $f(x, y) \to f(y, x)$ on the subject $t = f(a, b)$. The rule is applied at top-position, therefore we have $\omega_{xl} = 1$, $\omega_{yl} = 2$, $\omega_{xr} = 2$, $\omega_{yr} = 1$, $\dot{t} = f(a, b)$ and $\dot{r} = f((\!|\overline{\omega_{yr}} \bullet \texttt{-2} \bullet (1 \bullet \epsilon \bullet \omega_{yl})|\!), (\!|\overline{\omega_{xr}} \bullet \texttt{-2} \bullet (1 \bullet \epsilon \bullet \omega_{xl})|\!)) = f(\texttt{-1} \bullet \texttt{-2} \bullet 1 \bullet 2, \texttt{-2} \bullet \texttt{-2} \bullet 1 \bullet 1)$. Starting from $\langle f(a, b), \dot{r} \rangle$, we get $\langle f(a, b), \dot{r} \rangle_{[1 \leftrightarrow 2]} = \langle f(\texttt{-1} \bullet \texttt{-1} \bullet 2 \bullet 2, \texttt{-2} \bullet \texttt{-1} \bullet 2 \bullet 1), f(a, b) \rangle$. When computing the canonical form, we obtain $[\![\langle f(a, b), \dot{r} \rangle_{[1 \leftrightarrow 2]}]\!] = \langle f(b, a), f(\texttt{-1} \bullet \texttt{-2} \bullet 1 \bullet 1, \texttt{-2} \bullet \texttt{-2} \bullet 1 \bullet 2) \rangle$. Finally, the result is $[\![\langle f(a, b), \dot{r} \rangle_{[1 \leftrightarrow 2]}]\!]_{|1} = f(b, a)$ as expected.

The following example illustrates how collapsing rules are handled. Suppose we want to apply the rule $f(x) \to x$ to the subject $t = f(\texttt{-1})$. Since -1 is a path to the top of the redex, we have $\dot{t} = f(\texttt{-1} \bullet \texttt{-1} \bullet 2)$. In a second step $\dot{r}$ is evaluated to $\dot{r} = (\!|\texttt{-2} \bullet \texttt{deref}(\langle f(\texttt{-1} \bullet \texttt{-1} \bullet 2), x \rangle, 1 \bullet 1)|\!) = (\!|\texttt{-2} \bullet 2|\!) = \epsilon$ and we get $\langle f(\texttt{-1} \bullet \texttt{-1} \bullet 2), \epsilon \rangle_{[1 \leftrightarrow 2]} = \langle \epsilon, f(\texttt{-1} \bullet \texttt{-2} \bullet 1) \rangle$ which is already normalized. The result of the rewrite step is $\langle \epsilon, f(\texttt{-1} \bullet \texttt{-2} \bullet 1) \rangle_{|1} = \epsilon$ in accordance with [3].

By applying the rule $f(x, y) \to x$ to the first subterm, this last example shows what happens to pointers to subterms that disappear:



In this example, the subterm $g(b)$ is not preserved by the rule, therefore the reference to b is replaced by a copy of the subterm.

Theorem 2. *The set of ground canonical terms $\mathcal{T}_g(\mathcal{F})$ is closed under rewriting.*

Proof. Given a term $t \in \mathcal{T}_g(\mathcal{F})$, we have to prove that $[\![\langle \dot{t}, \dot{r} \rangle_{[1 \bullet \omega \leftrightarrow 2]}]\!]_{|1} \in \mathcal{T}_g(\mathcal{F})$. The ground term $\dot{r}$ is not a valid referenced term in itself because of the references that replace variables. The term $\dot{t}$ is neither valid because some paths can have been replaced by paths to the position 2. But, $\langle \dot{t}, \dot{r} \rangle$ is valid because invalid paths in $\dot{t}$ are now referencing the top position of $\dot{r}$ and invalid paths in $\dot{r}$ are referencing subterms of t. Since the swapping function preserves the notion of validity, the normal form $[\![\langle \dot{t}, \dot{r} \rangle_{[1 \bullet \omega \leftrightarrow 2]}]\!]$ exists. Thanks to the property of canonical referenced terms (pointers only from the right to the left), we can conclude that $[\![\langle \dot{t}, \dot{r} \rangle_{[1 \bullet \omega \leftrightarrow 2]}]\!]_{|1} \in \mathcal{T}_g(\mathcal{F})$. □

As the main result of the paper we show in the next section that every term-graph can be represented by a canonical referenced term and that term-graph rewriting (Definition 8) can be simulated by the algorithm introduced in Definition 17.

3.4 Simulation of term-graph rewriting

We introduce a function ϕ that translates any valid referenced term in $\mathcal{T}_{vr}(\mathcal{F},\mathcal{X})$ into a system of recursion equations (under the same set $\mathcal{F}$ and $\mathcal{X}$). For this purpose, we associate to every term $t \in \mathcal{T}_{vr}(\mathcal{F},\mathcal{X})$ a total function ψ_t from $\mathcal{P}os(t)$ to $\mathcal{X}$ defined as follows:

$$\psi_t(\omega) = \begin{cases} x & \text{if } \mathtt{symb}(t_{|\omega}) \in \mathcal{F} \cup \{\epsilon\} \\ & \text{where } x \text{ is a fresh variable} \\ t_{|\omega} & \text{if } t_{|\omega} \in \mathcal{X} \\ \psi_t(\omega') & \text{if } t_{|\omega} \in \mathcal{P}^* \\ & \text{where } \omega' = \mathtt{deref}(t,\omega) \end{cases}$$

Definition 18. *Given a valid referenced term $t \in \mathcal{T}_{vr}(\mathcal{F},\mathcal{X})$, we define its representation in systems of recursion equations by $\phi(t) = \{\alpha \mid \Delta\}$ where $\alpha = \psi_t(\epsilon)$ is the root, and Δ is a set of equation defined by:*

$$\begin{aligned} \Delta = \{\beta = f(\beta_1,\ldots,\beta_n) \mid\ & \omega \in \mathcal{P}os(t), \beta = \psi_t(\omega), \beta_i = \psi_t(\omega \centerdot i), \\ & \mathtt{symb}(t_{|\omega}) = f \in \mathcal{F}, arity(f) = n\} \\ \cup \{\gamma = \bullet \mid\ & \omega \in \mathcal{P}os(t), \gamma = \psi_t(\omega), \mathtt{symb}(t_{|\omega}) = \epsilon\} \end{aligned}$$

ϕ can be naturally extended to rules: $\phi(l \to r) = \phi(l) \to \phi(r)$.

Note that the equational representations of two equivalent valid referenced terms are equal modulo renaming. Moreover, the ϕ function is surjective so every system of recursion equations has an unique representation in $\mathcal{T}_g(\mathcal{F},\mathcal{X})$.

Theorem 3 (Rewrite step simulation). *Given a canonical referenced term $t \in \mathcal{T}_g(\mathcal{F})$ and a rule R, we have:*

$$t \to_R t' \Leftrightarrow \phi(t) \to_{\phi(R)} \phi(t')$$

Proof. The proof is in the Appendix B. □

Theorem 3 shows how to simulate term-graph rewriting using term-rewriting and provides a technique for easily extending rule-based languages with term-graphs and more generally with a notion of pointers.

Acyclic term graphs are widely used to obtain efficient term rewriting implementation [14]. On the contrary, our goal is not to improve the efficiency of term-rewriting engines but to offer a support for graph structure manipulation. The main objective of such extensions is to perform static analysis by rewriting control flow graphs or data-structures with pointers.

4 Related work

The notion of term graph has been intensively studied in the literature, in particular in [6], where a restricted form of *acyclic* term-graph has been used to represent terms with sharing. There is also a rich literature on modeling functional languages [7, 1, 2]. This approach leads to efficient implementations of functional languages or term rewriting engines, as in Clean, Elan, or Maude.

We are in a dual situation: we already have a very efficient term rewriting engine. This implementation is based on the notion of *maximally shared terms*, internally represented by acyclic graphs. Such a representation is purely functional and does not allow any side effect. This constraint makes the implementation of graphs and term-graphs difficult. The contribution of the paper is to provide a solution to represent and transform graphs in a functional environment. In addition to reuse efficient term structures, a main advantage of using a classical underlying term representation is to make possible the reuse of the notion of term rewriting strategy [16, 20] which allows control over how rules are applied.

The extension to cyclic term-graph rewriting has been studied and in particular linked up with rational term rewriting [8, 10]. Especially, a mapping from cyclic term-graph rewriting to rational parallel term rewriting can be defined. In this context, it is often difficult to deal with graph homomorphism. In this work, we choose to simulate term-graph rewriting as defined in [3] and to favor practical aspects. In this formalism, the matching corresponds to functional bisimulation. As a consequence, the pattern $g(\text{-}1)$ cannot be matched with the subject $g(g(\text{-}1 \centerdot \text{-}1))$ at the root position even if they both represent the same infinite term $g(g(g(\ldots)))$.

Moreover, the set of valid referenced terms where references are not interpreted as sharing but as oriented pointers (Definition 6) is to our knowledge a new approach that can be interesting to study and simulate object-oriented languages [12, 13]. For example, it could be used to model garbage collection algorithms. In the context of term-graph rewriting, an original approach is the formalism presented in [13] where the right-hand side of the rules consists in a set of actions on the pointers. The work presented in this paper is a first step towards an implementation of such a formalism.

4.1 Integration in a rewrite environment: the Tom language

As canonical referenced terms are terms, it is possible to extend in a non-intrusive way any rule-based language in order to support term-graph rewriting. The presented formalism is implemented in Tom and thus directly available at `http://tom.loria.fr`.

For now, it is possible to automatically generate from a signature the extended version for referenced terms where the normalization is integrated. As the Tom terms are implemented with maximal sharing, so are the term-graphs. This part of the implementation is presented in [5]. All the operations on paths have been also implemented. Thus users can define a system of term-graph rules

and it is automatically compiled in a basic Tom strategy based on the rewriting algorithm (Definition 17). These term-graph rewriting rules can then be integrated in a more complex strategy using Tom strategy combinators. As a consequence, all Tom features are available for term-graphs.

5 Conclusion

We have generalized the notion of term positions with *term paths* that are closely related to the notion of path in graphs and to the concept of de Bruijn indices [11]. By extending a signature with paths we obtained a new kind of terms called *referenced terms* which can contain pointers. This representation of pointers can be useful to model the semantics of imperative programming languages for instance.

In the second part of the paper, we introduced *canonical referenced terms* to represent term-graphs. As in the case of de Bruijn indices, the interest of this representation is to avoid problems of alpha conversion, compared to representations with labels or variables. Another advantage is that contrary to a recursion equation, the hierarchical structure of the term-graph is explicit because of its term representation. The last contribution of the paper is an original algorithm that simulates cyclic term-graph rewriting using canonical referenced terms. Thanks to pointers, the substitution can be applied in an unusual way, using swapping between the redex and the right-hand side of the rule.

To conclude, this formalism opens promising perspectives in terms of program transformation and code analysis. To the best of our knowledge, the integration in the Tom language constitutes actually one of the most active and maintained implementation of term-graph rewriting and thus provides a solid platform to experiment graph transformations in a concise and expressive way.

Acknowledgments: We are extremely grateful to Claude Kirchner who has been strongly involved in this work. He greatly contributed to the foundations of the paper. We also sincerely thank Rachid Echahed and Joe Wells for fruitful exchanges and remarks about preliminary versions of this work.

References

1. Z. M. Ariola and S. Blom. Cyclic lambda calculi. In *TACS '97: Third International Symposium on Theoretical Aspects of Computer Software*, LNCS, pages 77–106, 1997.
2. Z. M. Ariola and S. Blom. Skew confluence and the lambda calculus with letrec. *Annals of Pure and Applied Logic*, 117(1-3):95–168, 2002.
3. Z. M. Ariola and J. W. Klop. Equational term graph rewriting. *Fundamenta Informaticae*, 26(3/4):207–240, 1996.
4. F. Baader and T. Nipkow. *Term Rewriting and all That*. Cambridge University Press, 1998.
5. E. Balland and P. Brauner. Term-graph rewriting in tom using relative positions. In *TERMGRAPH'07: International Workshop on Computing with Terms and Graphs*, 2007.

6. H. P. Barendregt, M. van Eekelen, J. Glauert, J. Kennaway, M. Plasmeijer, and M. Sleep. Term graph rewriting. In *PARLE Parallel Architectures and Languages Europe*, volume 259 of *LNCS*, pages 141–158, 1987.
7. Z.-E.-A. Benaissa, P. Lescanne, and K. H. Rose. Modeling sharing and recursion for weak reduction strategies using explicit substitution. In *PLILP'96: Proceedings of the 8th International Symposium in Programming Languages: Implementations, Logics, and Programs*, volume 1140 of *LNCS*, pages 393–407, 1996.
8. A. Corradini and F. Drewes. (cyclic) term graph rewriting is adequate for rational parallel term rewriting. Technical Report TR-97-14, Dipartimento di Informatica, Pisa, Italy, 1997.
9. A. Corradini and F. Gadducci. Cpo models for infinite term rewriting. In *AMAST'95: Proceedings of the 4th International Conference in Algebraic Methodology and Software Technology*, volume 936 of *LNCS*, pages 368–384, 1995.
10. A. Corradini and F. Gadducci. Rational term rewriting. In *FoSSaCS '98: Proceedings of the First International Conference on Foundations of Software Science and Computation Structure*, volume 1378 of *LNCS*, pages 156–171, 1998.
11. N. G. de Bruijn. Lambda calculus notation with nameless dummies. a tool for automatic formula manipulation with application to the church-rosser theorem. *Indagationes Mathematicae*, 34:381–392, 1972.
12. D. J. Dougherty, P. Lescanne, and L. Liquori. Addressed term rewriting systems: Application to a typed object calculus. *Mathematical Structures in Computer Science*, 16:667–709, 2006.
13. R. Echahed and N. Peltier. Narrowing data-structures with pointers. In *ICGT'06: Proceedings of the Third International Conference on Graph Transformations*, volume 4178 of *LNCS*, pages 92–106, 2006.
14. B. Hoffmann and D. Plump. Implementing term rewriting by jungle evaluation. *RAIRO: Theoretical Informatics and Applications*, 25, 1991.
15. R. Kennaway. On graph rewritings. *TCS*, 52(1-2):37–58, 1987.
16. C. Kirchner. Strategic rewriting. In *International Workshop on Reduction Strategies in Rewriting and Programming - WRS*, volume 124 of *ENTCS*, 2004.
17. C. Kirchner and H. Kirchner. Rewriting, solving, proving. A preliminary version of a book is available at `www.loria.fr/~ckirchne/=rsp/rsp.pdf`, 1999.
18. M. Löwe. Algebraic approach to single-pushout graph transformation. *TCS*, 109(1–2):181–224, 1993.
19. D. Plump. *Handbook of Graph Grammars and Computing by Graph Transformation*, chapter Term graph rewriting, pages 3–61. World Scientific Publishing, 1999.
20. E. Visser. Stratego: A language for program transformation based on rewriting strategies. System description of Stratego 0.5. In *RTA'01: 12th International Conference on Rewriting Techniques and Applications*, volume 2051 of *LNCS*, pages 357–361, 2001.

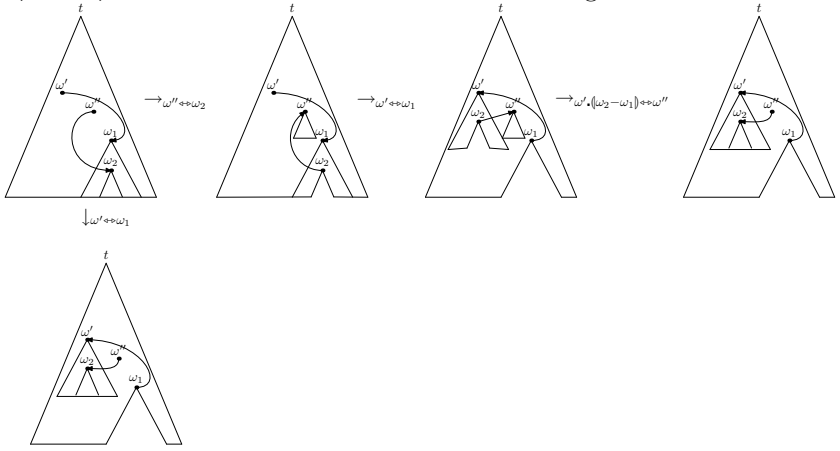
A Proof of convergence for the Definition 15

We have to prove termination and confluence. The termination property is trivial because of the total order on positions and of the fact that valid addressed terms do not contain pointers to pointers. Consider the multiset $\{\texttt{deref}(t,\omega) \mid t_{|\omega} \in \mathcal{P}\}$ composed of the pointed positions and the order on multisets (a multiset N is smaller than a multiset M if it is obtained by replacing finitely many elements of M by smaller elements). This multiset decreases strictly at each rewriting step. Indeed, after applying the rule at a given position, $t_{|\omega} \notin \mathcal{P}$ and $t_{|\texttt{deref}(t,\omega)} \in \mathcal{P}$. Now $\texttt{deref}(t,\texttt{deref}(t,\omega)) = \omega$ and $\omega <_{\mathcal{P}} \texttt{deref}(t,\omega)$. Concerning the other updated addresses, $\texttt{deref}(t,\omega)$ is either smaller (pointers to the swapped subterm or to subterms of it) or remains unchanged (pointers inside the subterm). Finally, the cardinality of the multiset has not evolved and each replaced element has been substituted by a smaller one so the termination is ensured.

Moreover, we will show that for the same reason, the normal form is unique. Suppose there are two distinct positions ω' and ω'' where the rule can be applied. We denote $\omega_1 = \texttt{deref}(t,\omega')$ and $\omega_2 = \texttt{deref}(t,\omega'')$. We distinguish several cases depending on the order between positions and the prefixation (noted $\sqsubseteq$). Some combinations of conditions are impossible because the rule can be applied only if $\omega' <_{\mathcal{P}} \omega_1$ and $\omega'' <_{\mathcal{P}} \omega_2$. Moreover, each prefixation $\omega \sqsubseteq \beta$ implies that $\omega <_{\mathcal{P}} \beta$. If we consider that $\omega' <_{\mathcal{P}} \omega''$, there are eight cases to consider:

$\omega'' < \omega_1 < \omega_2 \wedge \begin{cases} \omega_1 \sqsubseteq \omega_2 \\ \omega_1 \not\sqsubseteq \omega_2 \end{cases}$	$\omega'' < \omega_2 < \omega_1 \wedge \begin{cases} \omega_2 \sqsubseteq \omega_1 \\ \omega_2 \not\sqsubseteq \omega_1 \end{cases}$
$\omega' < \omega_1 < \omega'' < \omega_2 \wedge \begin{cases} \omega_1 \sqsubseteq \omega'' \wedge \omega_1 \sqsubseteq \omega_2 \\ \omega_1 \sqsubseteq \omega'' \wedge \omega_1 \not\sqsubseteq \omega_2 \\ \omega_1 \not\sqsubseteq \omega'' \wedge \omega_1 \not\sqsubseteq \omega_2 \end{cases}$	$\omega_1 = \omega_2$

For each case, we can prove local confluence. We will only detail the first case $\omega'' <_{\mathcal{P}} \omega_1 <_{\mathcal{P}} \omega_2 \wedge \omega_1 \sqsubseteq \omega_2$. The other cases are analogous.



If we start by swapping ω' and ω_1, as $\omega_1 \sqsubseteq \omega_2$, the subterm at position ω_2 is translated to the position $\omega'.(\!|\omega_2 - \omega_1|\!)$ and its pointer at position ω'' is updated.

Now $\texttt{deref}(t, \omega'') = \omega' \centerdot (\!|\omega_2 - \omega_1|\!)$ and as $\omega' <_{\mathcal{P}} \omega''$, $\texttt{deref}(t, \omega'') <_{\mathcal{P}} \omega''$, we do not need a second swapping.

If we start by swapping ω'' and ω_2, ω' is not updated and $\omega' <_{\mathcal{P}} \texttt{deref}(t, \omega') = \omega_1$. So we need to swap ω' and ω_1 and then translate the pointer at position ω_2 to the position $\omega' \centerdot (\!|\omega_2 - \omega_1|\!)$ which does not respect the order on positions $(\omega' \centerdot (\!|\omega_2 - \omega_1|\!) <_{\mathcal{P}} \texttt{deref}(t, \omega' \centerdot (\!|\omega_2 - \omega_1|\!)) = \omega'')$. We need a third swapping between ω'' and $\omega' \centerdot (\!|\omega_2 - \omega_1|\!)$.

As the normalization terminates, local confluence implies convergence. □

B Proof of Theorem 3

Theorem. *Given a canonical referenced term $t \in \mathcal{T}_g(\mathcal{F}, \mathcal{X})$ and a rule R, we have:*

$$t \to_R t' \Leftrightarrow \phi(t) \to_{\phi(R)} \phi(t')$$

First (I), we show that the notions of matching for systems of recursion equations and canonical referenced terms are equivalent. More formally, if we denote p the left-hand side of R, we have to prove that there exists a position ω such that $p \ll^{t,\omega}_{\epsilon} t_{|\omega} \parallel \emptyset$ reduces to $\top \parallel \Delta$ by application of $\mathcal{T}_g$-Matching if and only if there exists a variable substitution σ such that $\texttt{set}(\sigma(\phi(p))) \subseteq \texttt{set}(\phi(t))$. In (II), we will show that given t and t' such that $t \to_R t'$, the reduction of $\phi(t)$ by $\phi(R)$ is $\phi(t')$.

(I) ($\Rightarrow$) Let ω be a position ω such that $p \ll^{t,\omega}_{\epsilon} t_{|\omega} \parallel \emptyset$ reduces to $\top \parallel \Delta$, let σ be the relation $\{(\psi_p(\delta), \psi_t(\omega)) \mid (\delta, \omega) \in \Delta\}$. It is easy to show that σ is a functional bisimulation (the notion of bisimulation is explained in Definition 3.5 of [3]):

1. σ is a function from the variables of $\phi(p)$ to the variables of $\phi(t)$ because it is a relation with the appropriate domains and codomains by definition and because every rule of $\mathcal{T}_g$-Matching preserves the fact that for every pairs (δ, ω_1) and (δ, ω_2) in Δ, $\psi_t(\omega_1) = \psi_t(\omega_2)$,
2. the first step of rewriting ensures that $(\epsilon, \omega) \in \Delta$ so the root of $\sigma(p)$ is in relation with the root of the matched subterm $\psi_t(\omega)$,
3. the condition about congruence is respected due to the rule Decompose and because every pair of positions that appeared in a constraint has been added in Δ when the normal form $\top$ has been reached.

As a consequence we have $\texttt{set}(\sigma(\phi(p))) \subseteq \texttt{set}(\phi(t))$.

($\Leftarrow$) We now consider that there exists a variable substitution σ such that $\texttt{set}(\sigma(\phi(p))) \subseteq \texttt{set}(\phi(t))$. We need to show that there exists a position ω such that $p \ll^{t,\omega}_{\epsilon} t_{|\omega} \parallel \emptyset$ reduces to $\top \parallel \Delta$ by application of $\mathcal{T}_g$-Matching. Let ω be the position such that $\texttt{symb}(t_{|\omega}) \in \mathcal{F}$ and $\psi_t(\omega) = \texttt{root}(\sigma(\phi(p)))$. By construction of ψ_t and ϕ_t, ω exists and is unique.

To show that any reduction of $p \ll^{t,\omega}_{\epsilon} t_{|\omega} \parallel \emptyset$ leads to $\top$, we show that if the conjunction of matching constraints E is not reduced to $\top$, there is a rule

of $\mathcal{T}_g$-Matching that can be applied. Since $\mathcal{T}_g$-Matching is convergent, $\top$ is the normal form. Thanks to the convergence property proved in the Proposition 4, we can choose any strategy, in particular the one that consists in selecting the matching constraints which have the smallest δ according to $<_{\mathcal{P}}$. The considered matching constraint is unique because at each rewriting step, the conjunction E contains matching constraints with distinct δ.
We now consider two invariants that are used to show the expected result:

- Inv1: $\forall \delta, \omega\ (\delta, \omega) \in \Delta \cup \Gamma \Rightarrow (\psi_p(\delta), \psi_t(\omega)) \in \sigma$, where Γ is the set of pairs (δ', ω') that appear in the matching constraints of E
- Inv2: for every position $\delta' \in \mathcal{P}os(p)$ smaller than the smallest position δ in E, $\exists \omega'$ such that $(\delta', \omega') \in \Delta$

It is easy to show that Inv1 and Inv2 are invariant: they are true for the initial term $p \ll_{\epsilon}^{t,\omega} t_{|\omega} \parallel \emptyset$ and they are maintained by every rewrite step, for the considered strategy.

Now, we show that at each step of the derivation, a rule of $\mathcal{T}_g$-Matching can be applied. When considering the matching constraint $p' \ll_{\delta}^{t,\omega} t'$, we can distinguish four cases:

1. $\mathtt{symb}(p') \in \mathcal{F} \cup \{\epsilon\}$ and $\mathtt{symb}(t') \in \mathcal{F} \cup \{\epsilon\}$. In this case, $\mathtt{symb}(p') = \mathtt{symb}(t')$ because due to the invariant Inv1, $(\psi_p(\delta), \psi_t(\omega)) \in \sigma$ and as the substitution is a bisimulation, the property about congruence is respected. Decompose can be applied,
2. $\mathtt{symb}(p') \in \mathcal{X}$. We can apply the rule Variable,
3. $\mathtt{symb}(t') \in \mathcal{P}^*$. We can apply the rule Dereferencing,
4. $\mathtt{symb}(p') \in \mathcal{P}^*$ and $\mathtt{symb}(t') \in \mathcal{F} \cup \{\epsilon\}$. We need to show that $(\Vert \delta \mathtt{.symb}(\pi) \Vert, \omega) \in \Delta$ to apply the rule Stability. Thanks to Inv2 and as $\Vert \delta \mathtt{\ .\ symb}(\pi) \Vert <_{\mathcal{P}} \delta$ due to the Definition 13, there exists a pair $(\Vert \delta \mathtt{\ .\ symb}(\pi) \Vert, \omega') \in \Delta$. This pair is in Δ means that one of the four rules has been applied. Due to the Definition 13, we know that $\mathtt{symb}(p_{|\Vert \delta \mathtt{.symb}(\pi) \Vert}) \in \mathcal{F} \cup \{\epsilon\}$. Moreover, as $(\psi_p(\delta), \psi_t(\omega)) \in \sigma$ (Inv1) and $\psi_p(\Vert \delta \mathtt{\ .\ symb}(\pi) \Vert) = \psi_p(\delta)$ by definition of ψ, we know that $(\psi_p(\Vert \delta \mathtt{\ .\ symb}(\pi) \Vert), \psi_t(\omega)) \in \sigma$. As σ is a function, we have $\psi_t(\omega') = \psi_t(\omega)$. There are two cases:
   - the rule Decompose has been applied so $\mathtt{symb}(t_{|\omega'}) \in \mathcal{F}$. As $\mathtt{symb}(t_{|\omega'}) \in \mathcal{F}$ and $\psi_t(\omega') = \psi_t(\omega)$, $\omega' = \omega$.
   - the rule Dereferencing has been applied so the rule Decompose has been applied with the pair $(\Vert \delta \mathtt{\ .\ symb}(\pi) \Vert, \omega)$ and thus belongs to Δ.

$\mathcal{T}_g$-Matching is convergent. With the considered strategy, a matching constraint can always be reduced by a rule of $\mathcal{T}_g$-Matching. Therefore, the resulting normal form is $\top$.

(II) According to Definition 17, we have $t' = [\![\langle \dot{t}, \dot{r} \rangle_{[1 \mathbf{.} \omega \leftrightarrow 2]}]\!]_{|1}$, where r is the right-hand side of R. We want to show that for the system of recursion equations L such that $\phi(t) \rightarrow_{\phi(R)} L$ and $\psi_t(\omega) = \mathtt{root}(\sigma(\phi(p)))$ we have $L = \phi(t')$. According to Definition 8, we have $L = \{\alpha_t \mid (\mathtt{set}(\phi(t)) \setminus \{\alpha = \tau\}) \cup \mathtt{set}(\sigma'(\phi(r)))\}$, where $\alpha_t = \mathtt{root}(\phi(t))$, $\alpha = \mathtt{root}(\sigma(\phi(p)))$.

First $\phi(\langle \dot{t}, \dot{r} \rangle)$ can be expressed as $\{\alpha' \mid \Delta\}$ where $\Delta = \{\alpha' = \langle \alpha_t, \alpha \rangle\} \cup \mathtt{set}(\phi'(t)) \cup \mathtt{set}(\sigma'(\phi(r)))$. Let α'' be a fresh variable, $\phi'(t)$ corresponds to $\phi(t)$ where the equation $\alpha = t$ is replaced by $\alpha'' = t$ and if there exists a position β and a positive integer i such that $\omega = \beta \mathbin{.} i$, the equation $\psi_t(\beta) = f(\ldots, \alpha, \ldots)$ is replaced by $\psi_t(\beta) = f(\ldots, \alpha'', \ldots)$.

Now, we need to interpret the swapping operation in the context of equation systems. From the definitions of swapping and translation (Definitions 14 and 9), it is possible to derive $\phi(t_{[\omega_1 \leftrightarrow \omega_2]})$ from $\phi(t)$. Indeed, the swapping just consists in exchanging two subterms in t. $\phi(t_{[\omega_1 \leftrightarrow \omega_2]})$ is defined by:

- $\mathtt{root}(t_{[\omega_1 \leftrightarrow \omega_2]}) = \mathtt{root}(t)$,
- $\mathtt{set}(\phi(t_{[\omega_1 \leftrightarrow \omega_2]}))$ is $\mathtt{set}(\phi(t))$ where the equation $\beta_1 = f(\ldots, \alpha_1, \ldots)$ is replaced by $\beta_1 = f(\ldots, \alpha_2, \ldots)$ and the equation $\beta_2 = f(\ldots, \alpha_2, \ldots)$ is replaced by $\beta_2 = f(\ldots, \alpha_1, \ldots)$.
  $\alpha_1, \alpha_2, \beta_1, \beta_2$ are respectively $\psi_t(\omega_1), \psi_t(\omega_2), \psi_t(\pi_1), \psi_t(\pi_2)$ where $\omega_1 = \pi_1 \mathbin{.} i$, $\omega_2 = \pi_2 \mathbin{.} j$ and $i, j \in \mathbb{N}^*$.

By applying the swapping operation, we get $\phi(\langle \dot{t}, \dot{r} \rangle_{[\omega_1 \leftrightarrow \omega_2]}) = \{\alpha' \mid \{\alpha' = \langle \alpha_t, \alpha'' \rangle\} \cup (\phi(t) \setminus \{\alpha = \tau\}) \cup \{\alpha'' = \tau\} \cup \sigma'(\phi(r))\}$. As two equivalent valid referenced terms have the same representation by ϕ, we have $\phi(\llbracket \langle \dot{t}, \dot{r} \rangle_{[\omega_1 \leftrightarrow \omega_2]} \rrbracket) = \phi(\langle \dot{t}, \dot{r} \rangle_{[\omega_1 \leftrightarrow \omega_2]})$. In general, $\phi(t_{|\omega}) = \{\psi_t(\omega) \mid \Delta\}$ where Δ corresponds to the subset of $\mathtt{set}(\phi(t))$ of bounded variables reachable from $\psi_t(\omega)$. In our case $\omega = 1$, and thus the root is α_t. In addition, since there is no pointer to α' and α'', we can suppress their corresponding equations and we finally obtain $\phi(t') = \phi(\llbracket \langle \dot{t}, \dot{r} \rangle_{[\omega_1 \leftrightarrow \omega_2]} \rrbracket_{|1}) = \phi(\langle \dot{t}, \dot{r} \rangle_{[\omega_1 \leftrightarrow \omega_2]_{|1}}) = \{\alpha_t \mid (\mathtt{set}(\phi(t)) \setminus \{\alpha = \tau\}) \cup \mathtt{set}(\sigma'(\phi(r)))\}$ where equations corresponding to bounded variables unreachable from α_t have been cleaned.
This shows that given t and t' such that $t \to_R t'$, the reduction of $\phi(t)$ by $\phi(R)$ is $\phi(t')$. □