Redex capturing in term graph rewriting*

William M. Farmer and Ronald J. Watro

The MITRE Corporation
Burlington Road, A156
Bedford, MA 01730 USA
farmer@mitre.org, watro@mitre.org

November 1990

Abstract

Term graphs are a natural generalization of terms in which structure sharing is allowed. Structure sharing makes term graph rewriting a time- and space-efficient method for implementing term rewrite systems. Certain structure sharing schemes can lead to a situation in which a term graph component is rewritten to another component that contains the original. This phenomenon, called *redex capturing,* introduces cycles into the term graph which is being rewritten—even when the graph and the rule themselves do not contain cycles. In some applications, redex capturing is undesirable, such as in contexts where garbage collectors require that graphs be acyclic. In other applications, for example in the use of the fixed-point combinator Y, redex capturing acts as a rewriting optimization. We show, using results about infinite rewritings of trees, that term graph rewriting with arbitrary structure sharing (including redex capturing) is sound for left-linear term rewrite systems.

Keywords: Functional programming; Graph reduction; Infinite trees; Mathematical foundations; Term graphs; Term rewriting systems.

*This paper is a revised version of MITRE Technical Report M89-36 (July 1989). It has been accepted for presentation at RTA-91 and publication in the *International Journal of the Foundations of Computer Science*.

1 Introduction

Term rewriting is a model of computation and a model of formula manipulation that is employed in many areas of computer science, including abstract data types, functional programming, and automated reasoning. The most direct way of implementing term rewriting is via string rewriting or tree rewriting. However, these approaches are inefficient because there is no *structure sharing*, that is, it is not possible to represent multiple occurrences of a subterm by a single substring or subtree. A better approach to implementing term rewriting is through the use of a certain kind of graph rewriting. The basic idea is to represent terms as "term graphs" in which structure sharing is possible. Terms are then rewritten by rewriting their graphical representations. A single rewriting of a term graph can correspond to several rewritings of the term represented by the term graph, due to structure sharing. Thus term graph rewriting is a time-efficient as well as a space-efficient method for implementing term rewrite systems.

Just as a term rewrite rule is an ordered pair of terms, a term graph rewrite rule is an ordered pair of term graphs. We will define term graph rewriting in a manner slightly different from previous work. Our definition is nondeterministic in the sense that a redex can be reduced in several ways, depending on how structure is shared. (Other approaches (such as [1]) have assumed that the rewriting rule itself determines the structure sharing.) We use the nondeterministic approach because our goal is to prove a general soundness theorem that applies to all forms of structure sharing.

Term graphs without cycles are usually called simply *directed acyclic graphs (DAGs).* Proofs about acyclic systems are simplified by the fact that every finite DAG corresponds to a unique term. Staples [11, 12, 13] and Barendregt et al. [1] have studied graph rewriting without cycles. In [1], a strong soundness theorem is given, stating that when term rewriting is implemented by acyclic graph rewriting, then each result obtainable by graph rewriting is also obtainable by term rewriting. It is also shown in [1] that, in regular term rewrite systems, term graph rewriting is *complete* with respect to term rewriting.

Term graphs with cycles are useful for representing infinite objects and for efficiently implementing certain term rewrite rules, such as the rule for the fixed-point combinator Y (see Example 4.5). For some term graph rewrite rules, certain structure sharing schemes can lead to a situation in which a term graph component is rewritten to another component that contains the original. This phenomenon, called *redex capturing*, introduces cycles into the term graph being rewritten—even when the graph and the rule themselves do not contain cycles. Redex capturing can be undesirable in some situations. This is obviously the case in contexts where term graphs must be acyclic, such as in the presence of garbage collectors which only work correctly on acyclic graphs. Also, correctness proofs are complicated by the fact that cyclic term graphs do not correspond to unique finite terms. On the other hand, redex capturing acts as a very desirable rewriting optimization for the Y combinator. An interesting example of a system which illustrates the issues of structure sharing and redex capturing in term graph rewriting is the Clio verification system [2].

The nature of graph rewriting with cycles, and with redex capturing in particular, has not been adequately addressed in the literature. When cycles are allowed in term graph rewriting, it is no longer sound with respect to (finite) term rewriting in the strong sense described above. In this paper we give a careful description of what redex capturing is and how it occurs. We show that redex capturing results from the application of certain rewrite rules under certain structure sharing schemes. We also show that there is a natural sense in which term graph rewriting with redex capturing is a sound method for implementing left-linear term rewrite systems.

This latter result follows as a corollary from the main theorem of the paper, the Soundness Theorem, which says that, if only left-linear, left-finite, left-acyclic term graph rewrite rules are employed, a term graph rewriting of finite length with redex capturing corresponds to a certain infinite, but "convergent" tree rewriting. The Soundness Theorem makes no assumption about how structure sharing is performed. The proof of the theorem uses a lemma, called the $\omega\omega$-Lemma, which is the major technical result of the paper. From the Soundness Theorem we obtain as an immediate corollary a proof of the soundness of the cyclic rewrite rule for the fixed-point combinator Y. In an earlier paper, Farmer, Ramsdell and Watro [6] proved (without using infinite rewriting) that graph head reduction of combinator expressions using the cyclic Y-rule is a correct implementation of term head reduction. The soundness result proved here is a broad generalization of one part of the earlier correctness theorem.

The paper consists of five sections in addition to this introduction. In Sections 2 and 3, we present the basic definitions of term graphs and term graph rewriting. We emphasize that the nature of term graph rewriting is heavily dependent on the kind of structure sharing scheme that is employed. Section 4 is devoted to the phenomenon of redex capturing. We show that to avoid redex capturing there is a cost to be paid in terms of generality, implementation efficiency, or implementation simplicity. In Section 5, we discuss tree rewriting—which we view as a special case of term graph rewriting. Section 5 ends with a proof of the $\omega\omega$-Lemma. The Soundness Theorem and the corollaries described above are proved in Section 6.

2 Term graphs

We present in this section the definition of a term graph and several related concepts. A term graph is a natural generalization of a labeled tree. Informally, term graphs are rooted, directed, ordered graphs in which each node is labeled by a function symbol or a variable. Unlike trees, term graphs allow structure to be shared, and in particular, may contain cycles. Also, term graphs "unravel" into trees. Most of the definitions and propositions given below are minor adaptations of work of Barendregt et al. [1], Kennaway [7] and Raoult [10], . Unlike [1] and [10], we do allow term graphs to have more than one node labeled by the same variable.

Let $\mathcal{F}$ be a set of *function symbols* and let $\mathcal{V}$ be a set of *variables*. We assume that each function symbol f has a non-negative integer arity, denoted by $\mathrm{arity}(f)$, and we define $\mathrm{arity}(v) = 0$ for all variables v. For any set S, let S^* be the set of all finite words over S. The length of a word w is denoted by $|w|$.

A *term graph (over $\mathcal{F}$ and $\mathcal{V}$)* is a quadruple $(N, \mathrm{lab}, \mathrm{suc}, \rho)$ where N is a set of *nodes* with $\rho \in N$, $\mathrm{lab} : N \to \mathcal{F} \cup \mathcal{V}$, and $\mathrm{suc} : N \to N^*$ such that $|\mathrm{suc}(\alpha)| = \mathrm{arity}(\mathrm{lab}(\alpha))$ for all $\alpha \in N$. For $\alpha \in N$, $\mathrm{lab}(\alpha)$ is called the *label* of α and the members of $\mathrm{suc}(\alpha)$ are called the *successors* of α. ρ is called the *root* of the term graph. Let i-$\mathrm{suc}(\alpha) = \alpha_i$ for $\mathrm{suc}(\alpha) = \alpha_1 \cdots \alpha_n$. In the rest of this paper term graphs are simply called *graphs* and are denoted by G, H, etc. The components of a graph G are denoted by $N_G, \mathrm{lab}_G, \mathrm{suc}_G$, and ρ_G. When there is no possibility for confusion, the components of a graph G_s will be denoted by $N_s, \mathrm{lab}_s, \mathrm{suc}_s$, and ρ_s.

A *path in* a graph G is a finite sequence $p = \langle(\alpha_1, i_1), \ldots, (\alpha_n, i_n), \alpha_{n+1}\rangle$ $(n \geq 0)$ where $\alpha_1, \ldots, \alpha_{n+1} \in N_G$, $i_1, \ldots, i_n \in \omega$ (the natural numbers), and $\alpha_{m+1} = i_m$-$\mathrm{suc}(\alpha_m)$ for all m with $1 \leq m \leq n$. The path p is said to be *from α_1 to α_{n+1}* and to have *length n*. A graph is *root connected* if there is a path from the root to each node in the graph. The *depth of a node α* in G, written $\mathrm{dp}_G(\alpha)$, is the least $n \geq 0$ such that there is a path from ρ_G to α with length n. If $\alpha \in N_G$ and there is no path from ρ_G to α, $\mathrm{dp}_G(\alpha)$ is undefined. The *depth of a graph* G, written $\mathrm{dp}(G)$, is

$$\max\{\mathrm{dp}_G(\alpha): \ \alpha \in N_G \text{ and } \mathrm{dp}_G(\alpha) \text{ is defined}\}.$$

In many applications of graph rewriting, nodes with undefined depth are irrelevant and are removed (whenever possible). The task of detecting and removing such nodes is called *garbage collection*. In this paper, little attention will be given to nodes with undefined depth and the issue of garbage collection will be treated trivially.

A *cycle* is a path from a node to itself of length ≥ 1. A graph is *cyclic* if there is a cycle in the graph. A *tree* is a graph G such that there is exactly one path from the root of G to each other node of G. A *term* is a tree T such that N_T is finite.

Let G be a graph with $\beta \in N_G$. The *subgraph of G at β*, written G/β is the graph $(N', \mathrm{lab}', \mathrm{suc}', \beta)$ where

$$N' = \{\alpha \in N: \ \text{there is a path from } \beta \text{ to } \alpha\}$$

and lab' and suc' are the restrictions of lab and suc to N'. Notice that G/β is a root connected graph. An *extension* of G is a graph H such that $\rho_H = \rho_G$ and G is a subgraph of H.

A map $\varphi : N_G \rightarrow N_H$ is *homomorphic at $\alpha \in N_G$* if

(1) $\mathrm{lab}_H(\varphi(\alpha)) = \mathrm{lab}_G(\alpha)$ and

(2) $\mathrm{suc}_H(\varphi(\alpha)) = \bar{\varphi}(\mathrm{suc}_G(\alpha))$

where $\bar{\varphi}(\alpha_1 \cdots \alpha_n) = \varphi(\alpha_1) \cdots \varphi(\alpha_n)$. φ is a *homomorphism from G to H* if φ is homomorphic at every $\alpha \in N_G$. Notice that in a homomorphism there is no distinction between nodes labeled with 0-ary function symbols and nodes labeled with variables.

A homomorphism φ from G to H is *rooted* if $\varphi(\rho_G) = \rho_H$. A bijective homomorphism φ from G to H is an *isomorphism* from G to H. G and H are *equivalent*, written $G \approx H$, if there is a rooted isomorphism from G to H.

Proposition 2.1 *Let G be a root connected term graph.*

(1) For every $\alpha \in N_H$, there is at most one homomorphism φ from G to H with $\varphi(\rho_G) = \alpha$.

(2) A rooted homomorphism from a graph G' to G must be surjective.

(3) If there is a rooted homomorphism from G to a tree T, then $G \approx T$.

An *unraveling* of G is a tree T such that there is a rooted homomorphism from T to G/ρ_G. For each graph G, let $u(G)$ be some unraveling of G.

Proposition 2.2 *(1) For every G, $u(G)$ exists.*

(2) Any two unravelings of G are equivalent.

(3) A graph G is a tree iff $G \approx u(G)$.

G and H are *tree equivalent*, written $G \approx_{\mathrm{t}} H$, if $u(G) \approx u(H)$. A graph G is *consolidated* if no two distinct subgraphs of G are tree equivalent.

Proposition 2.3 *(1) $G \approx H$ implies $G \approx_{\mathrm{t}} H$.*

(2) If T and U are trees, then $T \approx_{\mathrm{t}} U$ implies $T \approx U$.

(3) For every graph G there is a consolidated graph G' such that $G \approx_{\mathrm{t}} G'$.

(4) If there is a rooted homomorphism from a consolidated graph G to a root connected graph H, then $G \approx H$.

A *weak homomorphism from G to H* is a map $\varphi : N_G \to N_H$ such that:

(1) φ is homomorphic at every $\alpha \in N_G$ with $\mathrm{lab}_G(\alpha) \in \mathcal{F}$.

(2) For all $\alpha, \beta \in N_G$ *with* $\mathrm{lab}_G(\alpha) = \mathrm{lab}_G(\beta) \in \mathcal{V}$,

$$H/\varphi(\alpha) \;\approx_{\mathrm{t}}\; H/\varphi(\beta).$$

Weak homomorphisms create bindings for domain nodes that are labeled by variables. Let φ_i be a weak homomorphism from G_i to H for $i = 1, 2$. φ_1 and φ_2 are *compatible* if, for all $\alpha_1 \in N_1$ and all $\alpha_2 \in N_2$ with $\mathrm{lab}_1(\alpha_1) = \mathrm{lab}_2(\alpha_2) \in \mathcal{V}$,

$$H/\varphi_1(\alpha_1) \;\approx_{\mathrm{t}}\; H/\varphi_2(\alpha_2).$$

When presenting specific graphs, we shall use a linear notation and sometimes a pictorial notation. The linear notation is derived from the linear notation for graphs given in [1]. However, we write $f^\alpha(-, \ldots, -)$ in place of $\alpha : f(-, \ldots, -)$; the expression f^α denotes a node α with label f. Also, the leading node-label pair in a linear notation corresponds to the root of the graph being described. With the pictorial notation, the root of a graph is always the top-most node, and nodes in the graph with undefined depth are not depicted. Otherwise, the pictorial notation is basically self-explanatory.

3 Term graph rewriting

A *(graph) rewrite rule* is a pair $r = \langle G_L, G_R \rangle$ of root connected graphs. G_L and G_R are called the *left side* and the *right side*, respectively, of the rewrite rule. r is a *term* [*tree*] *rewrite rule* if G_L and G_R are terms [trees]. r is *left-linear* if there are no distinct nodes $\alpha_1, \alpha_2 \in N_L$ with $\mathrm{lab}_L(\alpha_1) = \mathrm{lab}_L(\alpha_2) \in \mathcal{V}$. r is *left-finite* if N_L is finite. r is *left-acyclic* if G_L is acyclic.

A *redex in* a graph G is a pair $\Delta = (r, \theta)$ where r is a rewrite rule $\langle G_L, G_R \rangle$ and θ is a weak homomorphism from G_L to G. $\theta(\rho_L)$ is called the *root* of Δ. The *depth* of Δ *in* G, written $\mathrm{dp}_G(\Delta)$, is $\mathrm{dp}_G(\theta(\rho_L))$.

Let $\Delta = (\langle G_L, G_R \rangle, \theta_L)$ be a redex in G. To *reduce* Δ in G means to construct a new graph H from G in two steps as follows:

(1) *Build step.* Choose an extension H^* of G and a weak homomorphism θ_R from G_R to H^* such that θ_L and θ_R are compatible.

(2) *Redirection step.* For all $\alpha \in N_{H^*}$ and integers i such that $i\text{-suc}_{H^*}(\alpha) = \theta_L(\rho_L)$, redefine $i\text{-suc}_{H^*}(\alpha)$ to be $\theta_R(\rho_R)$. Also, if $\rho_{H^*} = \theta_L(\rho_L)$, redefine ρ_{H^*} to be $\theta_R(\rho_R)$. Let H be this new graph obtained from H^*.

Notice that the build step is nondeterministic but the redirection step is deterministic (once H^* and θ_R have been chosen). Suppose there is a redex $\Delta = (r, \theta_L)$ in G with root α. Then to *apply* r *at* α *in* G (*using* H^* and θ_R) means to reduce the redex Δ in G (using H^* and θ_R).

In our definition we have not specified any relationship between G and the image of G_R under θ_R. Thus the image of G_R is allowed to share the structure of G. There are several schemes for controlling structure sharing between G and the image of G_R. Four structure sharing schemes are briefly described below:

(1) *No structure sharing.* The simplest scheme is to allow no structure sharing at all between G and the image of G_R. When G is a term and $\langle G_L, G_R \rangle$ is a term rewrite rule, redex reduction under this scheme is the same as redex reduction in ordinary term rewriting. Of course, any implementation of rewriting based on this scheme will be highly inefficient.

(2) *Minimal structure sharing.* A very natural way of sharing structure between G and the image of G_R is to require that the following condition holds. Suppose $\alpha \in N_R$ and there is some $\beta \in N_L$ such that $\mathrm{lab}_R(\alpha) = \mathrm{lab}_L(\beta) \in \mathcal{V}$. Then $\theta_R(\alpha) = \theta_L(\gamma)$ for some $\gamma \in N_L$ such that $\mathrm{lab}_L(\gamma) = \mathrm{lab}_R(\alpha)$. Intuitively, this means that the image of a node in

G_R labeled with the variable x is equal to the image of some node in G_L labeled with x (provided such a node in G_L exists). This mode of structure sharing is incorporated in nearly all structuring sharing schemes used in implementations of term graph rewriting.

(3) *Maximal structure sharing.* Another scheme is to have maximal structure sharing between G and the image of G_R. H^* is chosen so that H is consolidated relative to G, i.e., if $\alpha \in N_H - N_G$, then there is no $\beta \in N_G \cup N_H$ distinct from α such that $H/\beta \approx_{\mathrm{t}} H/\alpha$. (Maximal structure sharing is similar to *hash consing* in some pure Lisp implementations.) Redex reduction based on this scheme can be costly to implement because it requires global information about the structure of G.

(4) *Rule-based structure sharing.* The final scheme that we shall discuss is presented in [1]. In this scheme, structure sharing is determined entirely by structure sharing between G_L and G_R. θ_R must satisfy two conditions: (1) if $\alpha \in N_L \cap N_R$, then $\theta_R(\alpha) = \theta_L(\alpha)$, and (2) if $\alpha \in N_R - N_L$, then $\theta_R(\alpha) \in N_{H^*} - N_G$. This scheme leads to less structure sharing than the former scheme, but it requires only local information about the structure of G.

Example 3.1 Suppose we would like to apply the rewrite rule

$\langle h(x^\alpha), g(x^\alpha, x^\alpha)\rangle$

$$\begin{tikzcd}
h \arrow[d] & \Rightarrow & g \arrow[d, bend right] \arrow[d, bend left] \\
x & & x
\end{tikzcd}$$

to the graph

$G = f(g^\beta(a^\gamma, a^\gamma), h(a^\gamma)).$

$$\begin{tikzcd}
& f \arrow[dl] \arrow[dr] & \\
g \arrow[dr, bend left] \arrow[dr, bend right] & & h \arrow[dl] \\
& a &
\end{tikzcd}$$

If we use the no structure sharing scheme above, the result is

$H_1 = f(g^\beta(a^\gamma, a^\gamma), g(a^\delta, a^\delta)) + h(a^\gamma).$

$$\begin{tikzcd}
& f \arrow[dl] \arrow[dr] & \\
g \arrow[d, bend left] \arrow[d, bend right] & & g \arrow[d, bend left] \arrow[d, bend right] \\
a & & a
\end{tikzcd}$$

If we use the minimal or rule-based structure sharing scheme, the result is

$$H_2 = f(g^{\beta}(a^{\gamma}, a^{\gamma}), g(a^{\gamma}, a^{\gamma})) + h(a^{\gamma}).$$

$$\begin{tikzcd}
& f \arrow[dl] \arrow[dr] & \\
g \arrow[dr, bend left] \arrow[dr, bend right] & & g \arrow[dl, bend right] \arrow[dl, bend left] \\
& a &
\end{tikzcd}$$

And, if we use the maximal structuring sharing scheme, the result is

$$H_3 = f(g^{\beta}(a^{\gamma}, a^{\gamma}), g^{\beta}) + h(a^{\gamma}).$$

$$\begin{tikzcd}
f \arrow[d, bend right] \arrow[d, bend left] \\
g \arrow[d, bend right] \arrow[d, bend left] \\
a
\end{tikzcd}$$

Our definition of redex reduction provides several opportunities for nodes with undefined depth to be introduced into H. For instance, in the example above, the node labeled with h in H_i $(i = 1, 2, 3)$ has undefined depth (and does not appear in the pictorial notation). In general, $\theta_L(\rho_L)$ will have undefined depth in H if $\theta_L(\rho_L) \neq \theta_R(\rho_R)$. In many applications of term graph rewriting, the data structure representing the node $\theta_L(\rho_L)$ is overwritten so that it represents the node $\theta_R(\rho_R)$. This provides a very efficient way of performing the redirection step of redex reduction.

Let $\mathcal{R}$ be a set of rewrite rules. An $\mathcal{R}$*-redex* in G is a redex (r, θ) in G where $r \in \mathcal{R}$. For root connected graphs G and H, $G \to_{\mathcal{R}} H$ means that $H = H'/\rho_{H'}$ where H' is the result of reducing an $\mathcal{R}$-redex Δ in G. We write $G \to_{\mathcal{R}}^{\Delta} H$ when we want to indicate the redex Δ. Let $\to_{\mathcal{R}}^{*}$ denote the reflexive and transitive closure of the relation $\to_{\mathcal{R}}$.

We shall assume in the rest of the paper that all rewrite rules are members of a fixed set $\mathcal{R}$ of rewrite rules. Since $\mathcal{R}$ is fixed we shall use $\to$, $\to^{\Delta}$, etc. as abbreviations for $\to_{\mathcal{R}}$, $\to_{\mathcal{R}}^{\Delta}$, etc.

A *graph rewriting* of a graph G_0 via $\mathcal{R}$ is either (1) a finite sequence $\Gamma = \langle (G_0, \Delta_0), \ldots, (G_n, \Delta_n), G_{n+1} \rangle$ $(n \geq 0)$ such that

$$G_0 \to^{\Delta_0} G_1 \to^{\Delta_1} G_2 \to^{\Delta_2} \cdots \to^{\Delta_n} G_{n+1}$$

or (2) an infinite sequence $\Gamma = \langle (G_0, \Delta_0), (G_1, \Delta_1), (G_2, \Delta_2), \ldots \rangle$ such that

$$G_0 \to^{\Delta_0} G_1 \to^{\Delta_1} G_2 \to^{\Delta_2} \cdots.$$

$$\begin{tikzcd} & & f \arrow[d] & & g \arrow[d] & & g \arrow[d] \\ r_1 = & x \Rightarrow & x & G_1 = & a & H_1 = & f \arrow[loop below] \end{tikzcd}$$

Figure 1: Example 4.1.

$$\begin{tikzcd} & f \arrow[d] & & & g \arrow[dl] \arrow[dr] & & & f \arrow[d] & & & g \arrow[dl] \arrow[loop right] \\ r_2 = & x & \Rightarrow & x & & f \arrow[d] & G_2 = & a & H_2 = & a & \\ & & & & & x & & & & & \end{tikzcd}$$

Figure 2: Example 4.2.

4 Redex capturing

Let $r = \langle G_L, G_R \rangle$ be a rewrite rule, $\Delta = (r, \theta_L)$ a redex in a graph G, H^* an extension of G, and θ_R a weak homomorphism from G_R to H^* such that θ_L and θ_R are compatible. Suppose that $\theta_L(\rho_L) \neq \theta_R(\rho_R)$ and that there exists a path from $\theta_R(\rho_R)$ to $\theta_L(\rho_L)$. If Δ is reduced using H^* and θ_R, the redirection step will create a cycle that begins and ends with $\theta_R(\rho_R)$. We call this phenomenon *redex capturing* (since the image of G_R "captures" the redex Δ). It is illustrated by the following three examples.

Example 4.1 [Figure 1] Let $r_1 = \langle x^\alpha, f(x^\alpha) \rangle$ and $G_1 = g(a^\beta)$. When r_1 is applied to the redex in G_1 with root β using either the minimal, maximal, or rule-based structure sharing scheme, the result is

$$H_1 = g(f^\gamma(f^\gamma)) + a^\beta.$$

Example 4.2 [Figure 2] Let $r_2 = \langle f^\alpha(x^\beta), g(x^\beta, f^\alpha) \rangle$ and $G_2 = f(a^\gamma)$. When r_2 is applied to G_2 using the maximal or rule-based structure sharing scheme, the result is

$$H_2 = g^\delta(a^\gamma, g^\delta) + f(a^\gamma).$$

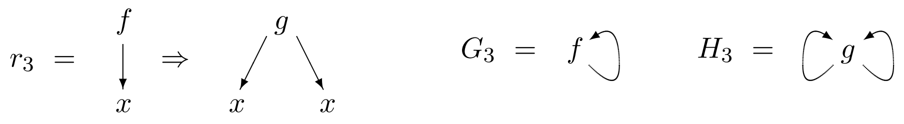

Figure 3: Example 4.3.

Example 4.3 [Figure 3] Let $r_3 = \langle f(x^\alpha), g(x^\alpha, x^\alpha)\rangle$ and $G_3 = f^\beta(f^\beta)$. When r_3 is applied to G_3 using either the minimal, maximal, or rule-based structure sharing scheme, the result is

$$H_3 = g^\gamma(g^\gamma, g^\gamma) + f^\beta(g^\gamma).$$

Examples 4.1 and 4.2 show that redex capturing can create cycles even when there are no cycles in the graph being rewritten or in the rewrite rule being applied. This means that, if cyclic graphs are undesired (e.g., so that simple garbage collectors can be used), redex capturing must be avoided. Example 4.3 shows that the application of the most innocuous sort of rewrite rule can, in the right situation, involve redex capturing.

Proposition 4.4 *Let $r = \langle G_L, G_R\rangle$ be a rewrite rule such that G_R contains a node labeled by a variable. Then there is some application of r involving redex capturing.*

Proof *Case 1.* $\mathrm{lab}_R(\rho_R) \notin \mathcal{V}$: Choose some $x \in \mathcal{V}$ such that there is some $\alpha_R \in N_R$ with $\mathrm{lab}_R(\alpha_R) = x$. Define $M_L = \{\alpha \in N_L : \mathrm{lab}_L(\alpha) = x\}$ and $M_R = \{\alpha \in N_R : \mathrm{lab}_R(\alpha) = x\}$. (Note: M_L could be empty.) For all $\alpha \in N_L$ and integers i such that $i\text{-suc}_L(\alpha) \in M_L$, redefine $i\text{-suc}_L(\alpha) = \rho_L$. Then let G be this new graph obtained from G_L. Similarly, for all $\alpha \in N_R$ and integers i such that $i\text{-suc}_R(\alpha) \in M_R$, redefine $i\text{-suc}_R(\alpha) = \rho_L$. Then let H^* be this new graph obtained from G_R and G. Define θ_L to be the rooted homomorphism from G_L to G and θ_R to be the rooted homomorphism from G_R to H^*. Since $\mathrm{lab}_R(\rho_R) \notin \mathcal{V}$, we may assume without loss of generality that $\rho_L \neq \rho_R$. Clearly, $\Delta = (r, \theta_L)$ is a redex in G, θ_L and θ_R are compatible, $\theta_L(\rho_L) \neq \theta_R(\rho_R)$, and H^* is an extension of G. Δ is captured when it is reduced in G_L using H^* and θ_R.

Case 2. $\mathrm{lab}_R(\rho_R) = v \in \mathcal{V}$: In this case, G_R contains just one node. We assume that there exists $\alpha_L \in N_L$ with $\mathrm{lab}_L(\alpha_L) = v$, for otherwise the proof is trivial. Given the assumption, the proof proceeds as in case one, but the construction of G includes a preliminary step that takes G_L and replaces some node labeled by v with another copy of G_L. The map θ_R then sends ρ_R to the root of the additional copy of G_L, to ensure that $\theta_L(\rho_L) \neq \theta_R(\rho_R)$. $\square$

Let us say that a rewrite rule $\langle G_L, G_R \rangle$ is *coherent* if there exists compatible weak homomorphisms θ_L and θ_R such that (1) the image of θ_L is acyclic and (2) there is a path from $\theta_R(\rho_R)$ to $\theta_L(\rho_L)$. Clearly, a rewrite rule $\langle G_L, G_R \rangle$ is coherent whenever G_L is an acyclic subgraph of G_R. (Thus r_1 and r_2 in Examples 4.1 and 4.2 are coherent.) The proof of Lemma 4.4 shows that, in cyclic graphs, redex capturing can occur with a very large class of rewrite rules. However, in acyclic graphs, redex capturing can only occur when coherent rewrite rules are applied with a certain amount of structure sharing. Hence, redex capturing can be avoided in acyclic graphs either by not using coherent rewrite rules or by using little or no structure sharing in the application of coherent rewrite rules. In Example 4.1, the latter prescription would mean not to use even the minimal structure sharing scheme in applying r_1 to G_1.

Another way to forestall the cycle generation effect of redex capturing is to redirect only those “arrows” that do not come from the image of the right side of the rewrite rule. This is done by employing the following alternative version of the redirection step of redex reduction:

> *Alternative redirection step.* For all $\alpha \in N_{H^*}$ and integers i such that (1) $i\text{-suc}_{H^*}(\alpha) = \theta_L(\rho_L)$ and (2) there is no path from $\theta_R(\rho_R)$ to α, redefine $i\text{-suc}_{H^*}(\alpha)$ to be $\theta_R(\rho_R)$. Also, if $\rho_{H^*} = \theta_L(\rho_L)$, redefine ρ_{H^*} to be $\theta_R(\rho_R)$. Let H be this new graph obtained from H^*.

If the alternative redirection step is used instead of the usual redirection step, no cycles will be generated as a result of redex capturing. (Of course, cycles could still be generated if cycles are present in the graph being reduced or in the right side of the rewrite rule being applied.) However, term graph rewriting with the alternative redirection step—*alternative term graph rewriting* for short—will generally be more expensive to implement than ordinary term graph rewriting since there must be a (sometimes costly) check to see if there are any paths from the root of the image of G_R to the root of the image of G_L. In conjunction with a structure sharing scheme, such as the maximal scheme, in which a significant effort is devoted to achieving a high degree of structure sharing, the cost of alternative term graph rewriting could be acceptable.

For many applications, redex capturing is definitely undesirable because it introduces cycles and possibly unwanted rewritings. However, for some applications, redex capturing can be used as an optimization technique. This is illustrated in the following example.

Example 4.5 [Figure 4] The rewrite rule for the fixed-point combinator Y is

$$r_Y = \langle A(Y,x), A(x, A(Y,x))\rangle.$$

This rule is interesting because the left side of the rule is a subgraph of the right side (thus the rule is coherent). A tempting alternative rule for Y, in which structure sharing is embodied, is

$$r'_Y = \langle A^\alpha(Y, x^\beta), A(x^\beta, A^\alpha)\rangle.$$

However, redex capturing results when this rule is applied using the rule-based structure sharing scheme. The effect is exactly equivalent to applying the cyclic rule

$$r^{\mathrm{c}}_Y = \langle A(Y,x), A^\alpha(x, A^\alpha)\rangle$$

using the minimal structure sharing scheme. In implementations of functional programming languages, r^{c}_Y is usually used instead of r_Y because it is computationally more efficient (for a discussion of this subject, see [9] and [14]). In particular, iterative procedures defined recursively with Y execute in constant space when r^{c}_Y is employed, but not when r_Y is used.

As we have shown above, redex capturing can be either desirable or undesirable, depending on the application that is involved. We have not, however, addressed the very important question of whether redex capturing ever leads to an unsound implementation of term rewriting. The rest of the paper is devoted to showing that term graph rewriting with redex capturing is in fact sound with respect to term rewriting, provided only left-linear term rewrite rules are employed (Corollary 6.3). We shall show this by proving that a finite rewriting with redex capturing corresponds to a (possibly infinite) "convergent tree rewriting". This result will be proved in Section 6, and the terminology and machinery behind "convergent tree rewritings" will be given in the next section.

5 Tree rewriting

In this section, we explore term graph rewriting restricted to trees and define a notion of convergence for infinite tree rewriting sequences. This subject

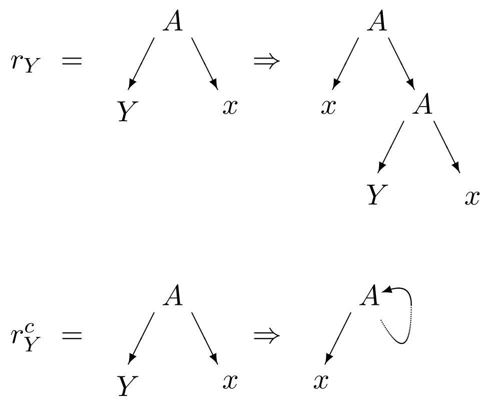

Figure 4: Example 4.5.

is also examined by Dershowitz, Kaplan and Plaisted [4], where a rewriting system $\mathcal{R}$ is called *top-terminating* if every ω-sequence of rewritings via $\mathcal{R}$ is convergent. Our notion of convergence is called *strong convergence* in the work of Kennaway, Klop, Sleep and de Vries [8].

Let T, U be trees and $d \geq 0$. Define $N_T^d = \{\alpha \in N_T : \mathrm{dp}_T(\alpha) \leq d\}$. A *rooted isomorphism from T to U to depth d* is a map $\varphi : N_T \to N_U$ such that (1) φ is homomorphic at each $\alpha \in N_T^d$ and (2) φ is a bijection from N_T^d to N_U^d. T is *equivalent to U to depth d*, written $T \approx_d U$, if there is a rooted isomorphism from T to U to depth d.

Let $\sigma = \langle T_0, T_1, T_2, \ldots \rangle$ be an infinite sequence of trees. The *limit* of σ, written $\lim(\sigma)$, is a tree T (which need not exist) such that

$$\forall d \geq 0\ \exists m \geq 0\ \forall n \geq m\ [T_n \approx_d T].$$

Define $u(\mathcal{R})$ to be the following set of tree rewrite rules:

$$\{\langle u(G_L), u(G_R) \rangle : \langle G_L, G_R \rangle \in \mathcal{R}\}.$$

Proposition 5.1 *If $G \to_{\mathcal{R}} H$, then $G \to_{u(\mathcal{R})} H$.*

Proof Suppose H is the result of reducing $(\langle G_L, G_R \rangle, \theta_L)$ in G using H^* and θ_R. Clearly, there are rooted surjective homomorphisms

$$\psi_L : u(G_L) \to G_L \ \text{ and } \ \psi_R : u(G_R) \to G_R.$$

The images of $u(G_L)$ and $u(G_R)$ under $\theta_L \circ \psi_L$ and $\theta_R \circ \psi_R$ are identical to the images of G_L and G_R under θ_L and θ_R, respectively. Also, $\Delta = (\langle u(G_L), u(G_R)\rangle, \theta_L \circ \psi_L)$ is a $u(\mathcal{R})$-redex in G. Therefore, H is the result of reducing Δ in G using H^* and $\theta_R \circ \psi_R$.$\Box$

A *tree rewriting via* $\mathcal{R}$ is a graph rewriting via $u(\mathcal{R})$ composed entirely of trees. For an infinite tree rewriting $\Gamma = \langle (T_0, \Delta_0), (T_1, \Delta_1), (T_2, \Delta_2), \ldots\rangle$, define $\sigma(\Gamma) = \langle T_0, T_1, T_2, \ldots\rangle$. $\rightarrow_{\mathrm{t},\mathcal{R}}$ is the relation $\rightarrow_{u(\mathcal{R})}$ restricted to trees, and $\rightarrow^*_{\mathrm{t},\mathcal{R}}$ is the reflexive and transitive closure of $\rightarrow_{\mathrm{t},\mathcal{R}}$. As before, we shall use $\rightarrow_{\mathrm{t}}$, $\rightarrow^{\Delta}_{\mathrm{t}}$, etc. as abbreviations for $\rightarrow_{\mathrm{t},\mathcal{R}}$, $\rightarrow^{\Delta}_{\mathrm{t},\mathcal{R}}$, etc.

Let Γ be a tree rewriting. If $\Gamma = \langle (T_0, \Delta_0), \ldots, (T_n, \Delta_n), T_{n+1}\rangle$, Γ *converges to* T means $T = T_{n+1}$. If $\Gamma = \langle (T_0, \Delta_0), (T_1, \Delta_1), (T_2, \Delta_2), \ldots\rangle$, Γ *converges to* T means that

$$T = \lim(\sigma(\Gamma)) \ \text{ and } \ \lim_{n\to\infty} \mathrm{dp}_{T_n}(\Delta_n) = \infty.$$

Proposition 5.2 *Let* $\Gamma = \langle (T_0, \Delta_0), (T_1, \Delta_1), (T_2, \Delta_2), \ldots\rangle$ *be an infinite tree rewriting. If* $\lim_{n\to\infty} \mathrm{dp}_{T_n}(\Delta_n) = \infty$, *then* $\lim(\sigma(\Gamma))$ *exists.*

Let $T \rightarrow^{\omega}_{\mathrm{t}} U$ mean that there is a tree rewriting of T which converges to U. For an integer $d \geq 0$, let $T \rightarrow^{d,*}_{\mathrm{t}} U$ mean that there is a tree rewriting $\Gamma = \langle (T_0, \Delta_0), \ldots, (T_n, \Delta_n), T_{n+1}\rangle$ such that $d \leq \mathrm{dp}_{T_i}(\Delta_i)$ for all i with $0 \leq i \leq n$. $T \rightarrow^{d,\omega}_{\mathrm{t}} U$ is defined similarly.

Lemma 5.3 ($\omega\omega$-Lemma) *Assume that each member of* $\mathcal{R}$ *is left-linear, left-finite, and left-acyclic.*

(1) If $T_0 \rightarrow^{\omega}_{\mathrm{t}} T_1 \rightarrow_{\mathrm{t}} T_2$, *then* $T_0 \rightarrow^{\omega}_{\mathrm{t}} T_2$.

(2) If $T_0 \rightarrow^{\omega}_{\mathrm{t}} T_1 \rightarrow^{\omega}_{\mathrm{t}} T_2$, *then* $T_0 \rightarrow^{\omega}_{\mathrm{t}} T_2$.

(3) Let $\sigma = \langle T_0, T_1, T_2, \ldots\rangle$ *be an infinite sequence of trees such that*

- *(a)* $T_0 \rightarrow^{d_0,\omega}_{\mathrm{t}} T_1 \rightarrow^{d_1,\omega}_{\mathrm{t}} T_2 \rightarrow^{d_2,\omega}_{\mathrm{t}} \cdots$.
- *(b)* $\lim_{n\to\infty} d_n = \infty$.

Then $T_0 \rightarrow^{\omega}_{\mathrm{t}} \lim(\sigma)$.

Before proving this lemma, we present three examples, one remark, and one lemma. The three examples show that the hypotheses of the the $\omega\omega$-Lemma cannot be eliminated.

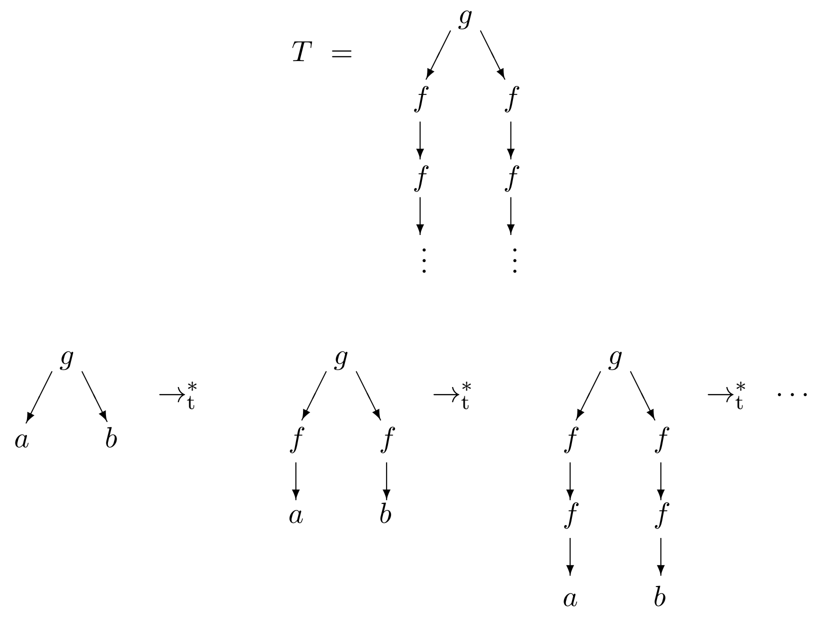

Figure 5: Example 5.4.

Example 5.4 [Figure 5] Let $T = u(g(f^{\alpha}(f^{\alpha}), f^{\beta}(f^{\beta})))$ and

$$\mathcal{R} = \{\langle a, f(a)\rangle, \langle b, f(b)\rangle, \langle g(x,x), c\rangle\}.$$

Clearly, $g(a,b) \to_{\mathrm{t}}^{\omega} T$ and $T \to_{\mathrm{t}} c$, but it is not the case that $g(a,b) \to_{\mathrm{t}}^{\omega} c$. This shows that the left-linear hypothesis of the $\omega\omega$-Lemma cannot be discarded. (This example is taken from [3].)

Example 5.5 [Figure 6] Let $T = u(f^{\alpha}(f^{\alpha}))$ and $\mathcal{R} = \{\langle x, f(x)\rangle, \langle T, b\rangle\}$. Clearly, $a \to_{\mathrm{t}}^{\omega} T$ and $T \to_{\mathrm{t}} b$, but it is not the case that $a \to_{\mathrm{t}}^{\omega} b$. This shows that the left-finite and left-acyclic hypotheses of the $\omega\omega$-Lemma cannot be discarded.

Example 5.6 [Figure 7] Let $T_1 = u(g(f^{\alpha}(f^{\alpha}), b))$, $T_2 = u(g(f^{\alpha}(f^{\alpha}), c))$, and $\mathcal{R} = \{\langle g(x,b), g(f(x),b)\rangle, \langle b, c\rangle\}$. Consider the infinite tree rewriting $\Gamma = \langle (g(a,b), \Delta_0), (g(f(a),b), \Delta_1), (g(f(f(a)),b), \Delta_2), \ldots\rangle$. Clearly, the depth of each redex Δ_i in its respective tree is 0, and so Γ does not converge. However, $\lim(\sigma(\Gamma)) = T_1$, and $T_1 \to_{\mathrm{t}} T_2$, but there is no infinite tree rewriting Γ' beginning with $g(a,b)$ such that $\sigma(\Gamma') = T_2$. Hence the $\omega\omega$-Lemma would not be true if $T \to_{\mathrm{t}}^{\omega} U$ were defined without the condition on the depth of redexes.

$$\begin{tikzcd}
T \; = & f \arrow[d] & a & \rightarrow_{\mathrm{t}} & f \arrow[d] & \rightarrow_{\mathrm{t}} & f \arrow[d] & \rightarrow_{\mathrm{t}} & \cdots \\
& f \arrow[d] & & & a & & f \arrow[d] & & \\
& \vdots & & & & & a & &
\end{tikzcd}$$

Figure 6: Example 5.5.

$$\begin{tikzcd}
T_1 \; = & & g \arrow[dl] \arrow[dr] & & T_2 \; = & & g \arrow[dl] \arrow[dr] & \\
& f \arrow[d] & & b & & f \arrow[d] & & c \\
& f \arrow[d] & & & & f \arrow[d] & & \\
& \vdots & & & & \vdots & &
\end{tikzcd}$$

$$\begin{tikzcd}
\Gamma \; = & & g \arrow[dl] \arrow[dr] & & \rightarrow_{\mathrm{t}} & & g \arrow[dl] \arrow[dr] & & \rightarrow_{\mathrm{t}} & & g \arrow[dl] \arrow[dr] & & \rightarrow_{\mathrm{t}} & \cdots \\
& a & & b & & f \arrow[d] & & b & & f \arrow[d] & & b & & \\
& & & & & a & & & & f \arrow[d] & & & & \\
& & & & & & & & & a & & & &
\end{tikzcd}$$

Figure 7: Example 5.6.

Remark 5.7 An incorrect version of the $\omega\omega$-Lemma (named Theorem 1) appeared in [3] and [5]. Example 5.6 is a counterexample to that result. A corrected version, generalized to sequences of arbitrary length, is given in [4]. The Compressing Lemma of [8] is another generalization of the $\omega\omega$-Lemma to sequences of arbitrary length.

Lemma 5.8 *Let $S \rightarrow_{\mathrm{t}}^{d,\omega} T \rightarrow_{\mathrm{t}}^{\Delta} T'$ where $\Delta = (\langle T_L, T_R\rangle, \theta)$ such that $\langle T_L, T_R\rangle$ is left-linear and left-finite, $d_1 = \mathrm{dp}_T(\Delta)$ and $d_2 = \mathrm{dp}(T_L)$. If $d_1 + d_2 < d$, then there is a redex $\Delta' = (\langle T_L, T_R\rangle, \theta')$ in S and a tree S' such that*

$$S \rightarrow_{\mathrm{t}}^{\Delta'} S' \rightarrow_{\mathrm{t}}^{\omega} T'.$$

Proof Since $\langle T_L, T_R\rangle$ is left-linear and left-finite and $d_1 + d_2 < d$, there is a redex $\Delta' = (\langle T_L, T_R\rangle, \theta')$ in S such that $\theta'(\rho_L) = \theta(\rho_L)$. Consider the set $\mathcal{U}$ of subtrees of S at nodes α where $\mathrm{dp}_S(\alpha) = d$. It follows from $S \rightarrow_{\mathrm{t}}^{d,\omega} T$ that the rewriting from S to T can be viewed as an interleaving of convergent rewriting of the members of $\mathcal{U}$. Since $d_1 + d_2 < d$, $\mathrm{dp}_S(\theta'(\beta)) < d$ for each $\beta \in N_L$ with $\mathrm{lab}_L(\beta) \in \mathcal{V}$. Consequently, in S' there are 0 or more isomorphic copies of each member of $\mathcal{U}$. For each copy there is a convergent rewriting corresponding to the convergent rewriting of the original. Therefore, a rewriting of S' converging to T' can be constructed by simply interleaving the rewritings of the copies of members of $\mathcal{U}$.$\square$

Proof of $\omega\omega$-Lemma. *Proof of (1):* The proof is trivial if $T_0 \rightarrow_{\mathrm{t}}^{*} T_1$; so assume this is not the case. Suppose $S \rightarrow_{\mathrm{t}}^{\omega} T \rightarrow_{\mathrm{t}}^{\Delta} U$ where $\Delta = (\langle T_L, T_R\rangle, \theta)$. The assumption on $\mathcal{R}$ implies that $\langle T_L, T_R\rangle$ is left-linear and left-finite. Let $d_1 = \mathrm{dp}_T(\Delta)$, $d_2 = \mathrm{dp}(T_L)$, and $d > d_1 + d_2$. By the definition of the convergence of a tree rewriting, there is some tree S' such that $S \rightarrow_{\mathrm{t}}^{*} S' \rightarrow_{\mathrm{t}}^{d,\omega} T$. Hence, by Lemma 5.8, $S' \rightarrow_{\mathrm{t}}^{\omega} U$, and so $S \rightarrow_{\mathrm{t}}^{\omega} U$.

Proof of (2): The proof is trivial if $T_0 \rightarrow_{\mathrm{t}}^{*} T_1$ and follows immediately from part (1) of the lemma if $T_1 \rightarrow_{\mathrm{t}}^{*} T_2$; so assume neither situation is the case. It is sufficient to prove that if

$$(*) \quad S \rightarrow_{\mathrm{t}}^{*} S' \rightarrow_{\mathrm{t}}^{d,\omega} T \rightarrow_{\mathrm{t}}^{d,\omega} U$$

then there is some $d' > d$ and some trees S'' and T' such that

$$(**) \quad S' \rightarrow_{\mathrm{t}}^{*} S'' \rightarrow_{\mathrm{t}}^{d',\omega} T' \rightarrow_{\mathrm{t}}^{d',\omega} U.$$

Suppose that $(*)$ is true and $d' > d$. By the definition of the convergence of a tree rewriting, there is some tree T' such that $T \to_{\mathrm{t}}^{*} T' \to_{\mathrm{t}}^{d',\omega} U$. From $S' \to_{\mathrm{t}}^{\omega} T \to_{\mathrm{t}}^{*} T'$ we obtain $S' \to_{\mathrm{t}}^{\omega} T'$ by a finite number of applications of part (1) of the lemma. Again by the definition of convergence, there is some tree S'' such that $S' \to_{\mathrm{t}}^{*} S'' \to_{\mathrm{t}}^{d',\omega} T'$. It follows immediately from these results that $(**)$ is true.

Proof of (3): $\lim(\sigma)$ exists because $\lim_{n\to\infty} d_n = \infty$ and $S \to_{\mathrm{t}}^{d,\omega} T$ implies $S \approx_d T$. It follows from the hypothesis and part (2) of the lemma that, for all $d \geq 0$, there is some $m \geq 0$ and some tree T such that $T_0 \to_{\mathrm{t}}^{*} T \to_{\mathrm{t}}^{d,\omega} T_m$ and, for all $n \geq m$, $d_n \geq d$. This implies that $T_0 \to_{\mathrm{t}}^{\omega} \lim(\sigma)$.□

6 Soundness of term graph rewriting

Two redexes $(\langle G_1, H_1\rangle, \theta_1)$ and $(\langle G_2, H_2\rangle, \theta_2)$ in G are *independent in G* if

$$\{\theta_1(\alpha) : \alpha \in N_{G_1}\} \cap \{\theta_2(\alpha) : \alpha \in N_{G_2}\} = \emptyset.$$

Lemma 6.1 *Assume that each member of $\mathcal{R}$ is left-linear, left-finite, and left-acyclic. If $G \to H$, then $u(G) \to_{\mathrm{t}}^{\omega} u(H)$.*

Proof Let $G \to^{\Delta} H$, where $\Delta = (\langle G_L, G_R\rangle, \theta)$. Define $d = \mathrm{dp}_G(\Delta)$.

Case 1. Δ is not captured: Let φ be a rooted homomorphism from $S = u(G)$ to G/ρ_G. Define $\mathcal{N}$ to be the largest subset of N_S such that:

(1) If $\alpha \in \mathcal{N}$, then $\varphi(\alpha) = \theta(\rho_L)$.

(2) If $\langle(\alpha_1, i_1), \ldots, (\alpha_n, i_n), \alpha_{n+1}\rangle$ is a path in S from $\alpha_1 = \rho_S$ to $\alpha_{n+1} \in \mathcal{N}$, then $\varphi(\alpha_i) \neq \theta(\rho_L)$ for all i with $1 \leq i \leq n$.

For each $\alpha \in \mathcal{N}$, there is a redex $\Delta_\alpha = (\langle u(G_L), u(G_R)\rangle, \theta_\alpha)$ in S such that θ_α is rooted weak homomorphism from $u(G_L)$ to an isomorphic copy of $u(G/\theta(\rho_L))$ at α. Then $\{\Delta_\alpha : \alpha \in \mathcal{N}\}$ is a set of independent redexes in $u(G)$ each with depth $\geq d$. Therefore, $u(G) \to_{\mathrm{t}}^{d,\omega} u(H)$ because Δ is not captured.

Case 2. Δ is captured: Since Δ is captured, there is an infinite graph rewriting given by

$$G = G_0 \to^{\Delta_0} G_1 \to^{\Delta_1} G_2 \to^{\Delta_2} \cdots,$$

such that, for all $n \geq 0$, (1) Δ_n has the form $(\langle G_L, G_R \rangle, \theta_n)$, (2) Δ_n is not captured, and (3) $\lim_{n\to\infty} \mathrm{dp}_{G_n}(\Delta_n) = \infty$. Hence by Case 1 we have

$$u(G_0) \rightarrow_{\mathrm{t}}^{d_0,\omega} u(G_1) \rightarrow_{\mathrm{t}}^{d_1,\omega} u(G_2) \rightarrow_{\mathrm{t}}^{d_2,\omega} \cdots$$

with $\lim_{n\to\infty} d_n = \infty$. Also, $\lim(\sigma) = u(H)$ for

$$\sigma = \langle u(G_0), u(G_1), u(G_2), \ldots \rangle.$$

Therefore, by part (3) of the $\omega\omega$-Lemma, we have $u(G) \rightarrow_{\mathrm{t}}^{\omega} u(H)$. □

Theorem 6.2 (Soundness Theorem) *Assume that each member of $\mathcal{R}$ is left-linear, left-finite, and left-acyclic. If $G \rightarrow^* H$, then $u(G) \rightarrow_{\mathrm{t}}^{\omega} u(H)$.*

Proof Follows immediately from Lemma 6.1 and part (2) of the $\omega\omega$-Lemma.□

The corollary below shows that there is a natural sense in which term graph rewriting with redex capturing is a sound method for implementing left-linear term rewrite systems (i.e., term rewrite systems having only left-linear rewrite rules).

Corollary 6.3 *Assume that each member of $\mathcal{R}$ is a left-linear term rewrite rule, and let G and H be finite, acyclic graphs. If there is a finite graph rewriting of G to H via $\mathcal{R}$, then there is a finite tree rewriting of $u(G)$ to $u(H)$ via $\mathcal{R}$ composed entirely of terms.*

Proof By the Soundness Theorem, $u(G) \rightarrow_{\mathrm{t}}^{\omega} u(H)$. Since G and H are finite, acyclic graphs, $u(G)$ and $u(H)$ are terms. By assumption, each member of $\mathcal{R}$ is a term rewrite rule. Therefore, any tree rewriting of $u(G)$ which converges to $u(H)$ must be finite and composed entirely of terms.□

The following example (which is closely related to Example 5.4) shows that the left-linear hypothesis in Corollary 6.3 cannot be discarded:

Example 6.4 [Figure 8] Let $\mathcal{R} = \{\langle a, f(a)\rangle, \langle b, f(b)\rangle, \langle g(x,x), c\rangle\}$. Then $g(a,b) \rightarrow^* g(f^\alpha(f^\alpha), f^\beta(f^\beta))$ (where redex capturing has occurred twice), and hence $g(a,b) \rightarrow^* c$. However, there is no rewriting of $g(a,b)$ to c composed entirely of terms.

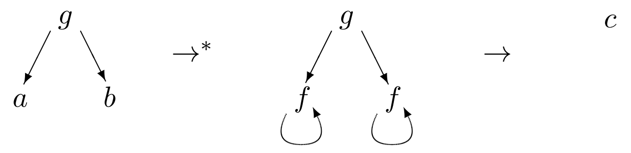

Figure 8: Example 6.4.

Corollary 6.5 *Let $\mathcal{R}$ be a set of combinator rewrite rules. The cyclic rewrite rule r_Y^{c} for the fixed-point combinator Y (see Example 4.5) is sound in the sense that, if $G \to^* H$ using r_Y^{c}, then $u(G) \to_{\mathrm{t}}^{\omega} u(H)$ using r_Y instead of r_Y^{c}.*

Proof A combinator rewrite rule is ordinarily left-linear, left-finite, and left-acyclic, so $\mathcal{R}$ satisfies the hypothesis of the Soundness Theorem. In Example 4.5 we showed that applying the acyclic rewrite rule r_Y for Y with redex capturing has the same effect as applying the cyclic rewrite rule r_Y^{c} for Y (without redex capturing). The corollary follows immediately from the Soundness Theorem and this observation.□

We conclude this paper with a simple example illustrating Corollary 6.5.

Example 6.6 [Figure 9] Let t be the term $A(Y, A(A(S,x),y))$. By applying the cyclic Y-rule r_Y^{c} and the rule for the S combinator, the term t rewrites to the cyclic graph $A^\alpha(A(x,A^\alpha), A(y,A^\alpha))$. This graph is a normal form, meaning that no further rewriting is possible. When the rule r_Y is used without redex capturing, t rewrites in two steps to the term $A(A(x,t),A(y,t))$. Acyclic rewriting can continue indefinitely, establishing an infinite rewriting sequence which converges to the unraveling of the graph normal form.

Acknowledgments

The authors wish to express special thanks to Dr. Leonard Monk for many valuable criticisms and suggestions. The authors are also grateful to The MITRE Corporation for supporting this work under the MITRE-Sponsored Research program.

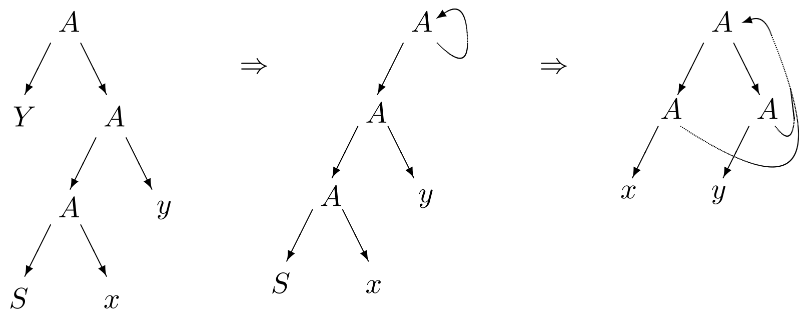

Figure 9: Example 6.6.

References

[1] H. P. Barendregt, M. C. J. D. van Eekelen, J. R. W. Glauert, J. R. Kennaway, M. J. Plasmeijer, and M. R. Sleep, "Term graph rewriting," in *PARLE – Parallel Architectures and Languages Europe*, *Springer Lecture Notes in Computer Science 259*, (Springer-Verlag, Berlin, 1987), pp. 141–158.

[2] M. Bickford, C. Mills, and E. A. Schneider, *Clio: An applicative language-based verification system*, Tech. Rep. 15-7, Odyssey Research Associates, Ithaca, NY, June 1989.

[3] N. Dershowitz and S. Kaplan, "Rewrite, rewrite, rewrite, rewrite, rewrite, ...," in *Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages*, 1989, pp. 250–259.

[4] N. Dershowitz, S. Kaplan, and D. A. Plaisted, "Rewrite, rewrite, rewrite, rewrite, rewrite, ...," to appear.

[5] N. Dershowitz, S. Kaplan, and D. A. Plaisted, "Infinite normal forms," in *Proceedings of the 16th International Colloquium on Automata, Languages, and Programming*, *Lecture Notes in Computer Science 372*, (Springer-Verlag, Berlin, 1989), pp. 249–262.

[6] W. M. Farmer, J. D. Ramsdell, and R. J. Watro, "A correctness proof for combinator reduction with cycles," *ACM Trans. Prog. Lang. Syst.* **12** (1990) 123–134.

[7] R. Kennaway, "On 'On graph rewritings'," *Theoret. Comp. Sci.* **52** (1987) 37–58.

[8] J. R. Kennaway, J. W. Klop, M. R. Sleep, F. J. de Vries, "Transfinite reductions in orthogonal term rewriting systems," to appear, Center for Mathematics and Computer Science (CWI) Report CS-R9042, Amsterdam, The Netherlands; also, to appear, *Proceedings of the Fourth International Conference on Rewriting Techniques and Applications*, April 1991.

[9] S. L. Peyton Jones, *The Implementation of Functional Programming Languages*, (Prentice Hall, New York, 1987).

[10] J. C. Raoult, "On graph rewritings," *Theoret. Comp. Sci.* **32** (1984) 1–24.

[11] J. Staples, "Computation on graph-like expressions," *Theoret. Comp. Sci.* **10** (1980) 171–185.

[12] J. Staples, "Optimal evaluations of graph-like expressions," *Theoret. Comp. Sci.* **10** (1980) 297–316.

[13] J. Staples, "Speeding up subtree replacement systems," *Theoret. Comp. Sci.* **11** (1980) 39–47.

[14] D. A. Turner, "A new implementation technique for applicative languages," *Soft. Pract. Exper.* **9** (1979) 31–49.