Graph Rewriting and Copying

Erik Barendsen* Sjaak Smetsers†
University of Nijmegen‡

Abstract

Graph rewrite systems play an important role in the implementation of functional programming languages. This paper focusses on the theoretical aspects of graph rewriting. Graph reduction is described in terms of general operations on graphs, leading to a substitution-like mechanism called *graph replacement*. Some basic theory on the interaction of these operations is developed. The ordering induced by *graph homomorphisms* is investigated. Equipped with this ordering, various sets of graphs have nice domain theoretical properties. Moreover, a *lazy copying* mechanism is added to the notion of graph reduction. This copying concept is analyzed by splitting it into more elementary graph operations, leading to a confluence result. Finally, some ideas are presented on the use of copying in practice.

*Supported by the Dutch organization for scientific research (NWO).

†Supported by the ESPRIT basic research action *SemaGraph*.

‡Department of Computer Science, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, e-mail `erikb@cs.kun.nl`, `sjakie@cs.kun.nl`, fax +31.80.652525.

Contents

1 Introduction 1

2 Graphs 2

3 Comparing graphs 4

4 Graph rewriting 12

5 Graph rewrite systems 20

6 Combining rewriting and unraveling 23

7 Copying 25

8 Analyzing copy reduction 28

9 Confluence 33

10 Towards an implementation of lazy copying 39

11 Conclusion 41

1. Introduction

There are several models of computation that can be viewed as a theoretical basis for functional programming.

The *lambda calculus* (see Barendregt (1984)), being a well-understood mathematical theory, is traditionally considered to be the purest foundation for functional languages. Later, the concept of *term rewrite systems* (TRS, see Klop (1992)) provided a theory that fits somewhat closer to actual functional programming.

A drawback of the abovementioned models of computation is the large gap between those models and the actual implementation of functional languages. E.g. when evaluating an expression of the form FA in λ-calculus or TRS, the argument A might get duplicated many times. This is far from efficient if A contains function applications that still need to be evaluated. Some optimizations of lambda calculus, involving *sharing* of arguments by different functions, have been proposed by Wadsworth (1971) and Lamping (1990).

People have sought for a model of computation that is sufficiently elegant and abstract, but at the same time incorporates mechanisms that are more realistic with respect to actual implementation techniques. Such a model could not only be used to study to study abstract properties of functional languages, but also as a tool to investigate the practical behaviour of functional programs such as complexity in time and in use of memory.

A promising candidate for such a model is provided by the notion of *graph rewrite systems* (GRS), introduced in Barendregt et al. (1987a). Graph rewriting combines the flexibility of TRS with the possibility of *sharing* expressions and building *cyclic* objects.

In Barendregt et al. (1987b), a restricted form of GRS is introduced: the so called *term graph rewrite systems* (TGRS). These TGRS are very well suited as a basis for implementation of functional languages. This is demonstrated by the 'intermediate language' CLEAN that is based on TGRS (see Brus et al. (1987)). Sequential implementations based on graph rewriting turned out to be successful (see Smetsers et al. (1991)).

The notion of TGRS is still very fresh; it lacks the tradition that can be found in fields such as lambda calculus and term rewrite systems. Taking Barendregt et al. (1987b) as a starting point, this paper tries to contribute to the development of the basic theory of term graph rewriting, from an operational point of view.

The sections 2, 4 and 5 contain a description of graph rewriting. Some new operations such as *segment*, *extension* and *replacement* are introduced, generalizing notions suggested by Barendregt et al. (1987b). A natural ordering on graphs is investigated in section 3.

One of the reasons for the impopularity of graph rewriting is the inherent complexity of the objects involved. Most operations on graphs cannot be decomposed into local actions, but have to be considered as a global action on each graph as a whole. This makes reasoning about graphs substantially more delicate than reasoning about terms; we therefore give many proofs in detail. This paper provides some general tools for reasoning about graphs in a convenient way.

Although pure TGRS's are convenient to model *sequential* computations, they do not capture some aspects that are important in *parallel* implementations. In a func-

tional program the evaluation of any expression is independent of the chosen reduction order by the absence of side effects. This makes functional languages attractive for being implemented on parallel hardware. An important class of parallel machines is formed by systems of loosely coupled processors in which each processing element is equipped with its own local memory. Exchange of information between processors takes place via some communication network. An implementation of functional languages on such a machine has to deal with the transfer of data during program execution. Hence the *communication* mechanism plays an important role.

In the 'overall view' of the computation the communication between two processes involves duplication of data. In order to investigate properties of such a computation, graph rewriting systems are extended with a copying mechanism. The information transfer between two processors is determined by the moment at which a processor needs data, and the amount of information that is available (i.e. not subject to a present computation) at that moment. In terms of copying this means that it should be possible to defer copying at some points in the graph that is being copied until these parts of the computation have been finished. This way of copying is called *lazy copying*, introduced in Eekelen et al. (1991). In section 7 some approaches to copying are discussed, and the notion of lazy copying is formalized, resulting in the notion of *copy term graph rewrite system* (C-TGRS). The rest of the paper is devoted to the investigation of some properties of C-TGRS, leading to the proof of a confluence result.

It turns out that the extension of TGRS's with copying can be described in an elegant way. It is also attractive from a theoretical point of view: such extended TGRS's provide an intermediate formalism between TRS's (where inputs of functions are duplicated during a rewrite step $\mathbf{F}(x) \to \mathbf{G}(x, x)$) and TGRS's (where this duplication does not take place).

2. Graphs

This section recalls some basic notions that were introduced by Barendregt et al. (1987b).

The objects of our interest are directed graphs in which each node is labeled with a specific symbol. The number of outgoing edges of a node is determined by its symbol. In the sequel we assume that $\mathcal{N}$ is some basic set of *nodes* (infinite; one often takes $\mathcal{N} = \mathbb{N}$), and Σ is a (possibly infinite) set of *symbols* with *arity* in $\mathbb{N}$.

2.1. Definition. (i) A *labeled graph* (over $\langle \mathcal{N}, \Sigma \rangle$) is a triple

$$g = \langle N, symb, args \rangle$$

such that

(1) $N \subseteq \mathcal{N}$; N is the set of *nodes* of g;
(2) $symb : N \to \Sigma$; $symb(n)$ is the *symbol* at node n;
(3) $args : N \to N^*$ such that

$$\text{length}(args(n)) = \text{arity}(symb(n)).$$

Thus $args(n)$ specifies the outgoing edges of n. The i-th component of $args(n)$ is denoted by $args(n)_i$.

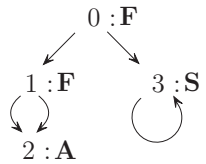
(ii) A *rooted graph* is a quadruple

$$g = \langle N, symb, args, r \rangle$$

such that $\langle N, symb, args \rangle$ is a labeled graph, and $r \in N$. The node r is called the *root* of the graph g.

(iii) The collection of all rooted labeled graphs over $\langle \mathcal{N}, \Sigma \rangle$ is indicated by GRAPHS. The set of finite rooted graphs is $\mathbb{G}$.

2.2. EXAMPLE. The following is a suggestive drawing of a rooted graph.



Here $N = \{0, 1, 2, 3\}$, $symb(0) = \mathbf{F}$, $symb(1) = \mathbf{F}$, etc., arity($\mathbf{F}$) = 2, arity($\mathbf{S}$) = 1, arity($\mathbf{A}$) = 0, $args(0) = (1, 3)$, $args(1) = (2, 2)$, etc., $r = 0$. The (names of) nodes are usually left implicit in drawings; they are only displayed when needed.

CONVENTION. (i) $m, n, n', \ldots$ range over nodes; $g, g', h, \ldots$ range over (rooted) graphs.

(ii) If g is a (rooted) graph, then its components are referred to as N_g, $symb_g$, $args_g$ (and r_g) respectively. Further abbreviations, such as writing N_2 instead of N_{g_2}, will be self-explanatory.

(iii) To simplify notation we usually write $n \in g$ instead of $n \in N_g$.

2.3. DEFINITION. (i) A *path* in a graph g is a sequence

$$p = (n_0, i_0, n_1, i_1, \ldots, n_\ell)$$

where $n_0, n_1, \ldots, n_\ell \in g$, and $i_0, i_1, \ldots, i_{\ell-1} \in \mathbb{N}$ are 'edge specifications' such that

$$n_{k+1} = args(n_k)_{i_k} \text{ for all } k < \ell.$$

In this case p is said to be a *path from* n_0 *to* n_ℓ (notation $p : n_0 \rightsquigarrow n_\ell$).

(ii) Let p be as above. The *length* of p (notation $|p|$) is ℓ. p *contains* n (notation $n \in p$) if $n_k = n$ for some $k \leq \ell$. p is a *rooted path* if it starts with the root of the graph (i.e. $p : r_g \rightsquigarrow n$ for some $n \in g$). The collection of rooted paths in g is denoted by RP(g). p is *cyclic* if $n_j = n_k$ for some $j \neq k$.

(iii) Let $m, n \in g$. m is *reachable from* n (notation $n \rightsquigarrow m$) if $p : n \rightsquigarrow m$ for some path p in g.

(iv) The *concatenation* of two paths p_1 and p_2 (notation $p_1 * p_2$) is defined in the obvious way, such that for $p_1 : n \rightsquigarrow n'$, $p_2 : n' \rightsquigarrow n''$ one has $p_1 * p_2 : n \rightsquigarrow n''$.

2.4. DEFINITION. Let g be a rooted graph.

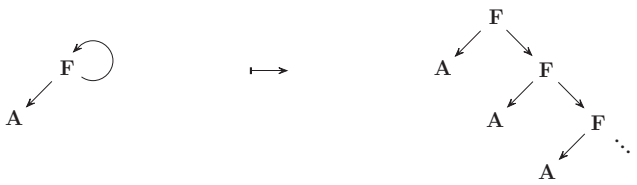
(i) g is *coherent* if

$$\forall n \in g \;\; \exists p \quad p : r_g \rightsquigarrow n.$$

(ii) g is a *tree* if

$$\forall n \in g \quad \exists! p \quad p : r_g \rightsquigarrow n.$$

To each graph g a (possibly infinite) tree can be assigned, by completely *unraveling* g. E.g.



2.5. Definition. Let g be a graph. The *complete unraveling* of g (notation $\mathrm{U}(g)$) is defined as follows. Set $\mathrm{U}(g) = u$, where

$$\begin{array}{rcll} N_u & = & \mathrm{RP}(g), & \\ symb_u(p) & = & symb_g(n) & \text{if } p : r_g \rightsquigarrow n, \\ args_u(p)_i & = & p * (n, i, args_g(n)_i) & \text{if } p : r_g \rightsquigarrow n, \\ r_u & = & (r_g). & \end{array}$$

2.6. Lemma. *Let $p_1, p_2 \in \mathrm{U}(g)$. If $args_u(p_1)_i = args_u(p_2)_j$, then $p_1 = p_2$ and $i = j$.*

Proof. Straightforward, by examining the definition of $\mathrm{U}(g)$. □

2.7. Lemma. *$\mathrm{U}(g)$ is a tree.*

Proof. By induction on the length of $p \in \mathrm{RP}(g)$ it can be shown that there is a path $q : r_u \rightsquigarrow p$ in $\mathrm{U}(g)$. Lemma 2.6 is used to show the uniqueness of such q. □

2.8. Definition. Let g be a graph and $n \in g$. The *subgraph of g at n* (notation $g \mid n$) is the rooted graph

$$\langle N, symb, args, n \rangle$$

where $N = \{m \in g \mid n \rightsquigarrow m\}$, and $symb$ and $args$ are the restrictions (to N) of $symb_g$ and $args_g$ respectively.

3. Comparing graphs

In this section we study an ordering $\preceq$ on graphs, induced by structure preserving mappings called graph homomorphisms. Equipped with this ordering, various sets of graphs have nice domain theoretical properties such as the existence of least upper bounds.

3.1. Definition. Let g, h be graphs, and $\varphi : N_g \to N_h$.

(i) φ is a *(graph) homomorphism* from g to h (notation $\varphi : g \to h$) if for all $n \in g$

$$\begin{array}{rcl} symb_h(\varphi(n)) & = & symb_g(n), \\ args_h(\varphi(n))_i & = & \varphi(args_g(n)_i). \end{array}$$

(ii) φ is a *rooted homomorphism* (notation $\varphi : g \xrightarrow[\mathrm{r}]{} h$) if $\varphi : g \to h$ and moreover $\varphi(r_g) = r_h$.

3.2. Remark. Let g be a coherent graph.

(i) If $\varphi : g \to h$ then φ is completely determined by $\varphi(r_g)$. Consequently, if $\varphi : g \xrightarrow[\mathrm{r}]{} h$ then φ is unique.

(ii) $\varphi : g \to h \;\Rightarrow\; \varphi : g \xrightarrow[\mathrm{r}]{} h \mid \varphi(r_g)$.

3.3. Definition. Let $\varphi : g \to h$, and let $p = (n_0, i_0, \ldots, n_k)$ be a path in g. Define

$$\varphi(p) = (\varphi(n_0), i_0, \ldots, \varphi(n_k)).$$

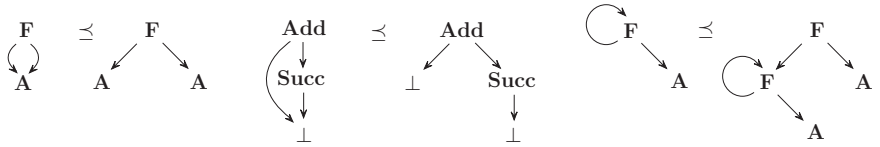
Then $\varphi(p)$ is a path in h.

3.4. Definition. The ordering $\preceq$ on rooted graphs is defined as follows.

$$g \preceq h \quad \text{if} \quad \varphi : h \xrightarrow[\mathrm{r}]{} g \text{ for some } \varphi.$$

Sometimes we write $g \preceq_\varphi h$ to indicate the homomorphism explicitly.

3.5. Example.



3.6. Lemma. (i) *The class of rooted homomorphisms is closed under composition, i.e.*

$$\varphi : g \xrightarrow[\mathrm{r}]{} g',\ \psi : g' \xrightarrow[\mathrm{r}]{} g'' \;\Rightarrow\; \psi \circ \varphi : g \xrightarrow[\mathrm{r}]{} g''.$$

(ii) $g \preceq g',\ g' \preceq g'' \;\Rightarrow\; g \preceq g''$.

Proof. (i) Trivial.

(ii) By (i). $\square$

3.7. Definition. Let g, h be graphs. Let $\varphi : N_g \to N_h$.

(i) φ is a *graph isomorphism* if $\varphi : g \to h$ and φ is bijective.

(ii) φ is a *rooted graph isomorphism* if $\varphi : g \xrightarrow[\mathrm{r}]{} h$ and φ is bijective.

3.8. Definition. (i) g is *isomorphic* to h (notation $g \cong h$) if there exists an isomorphism from g to h.

(ii) g is *equivalent* to h (notation $g \equiv h$) if there exists a rooted isomorphism from g to h.

3.9. Lemma. $g \preceq h,\ h \preceq g \;\Rightarrow\; g \equiv h$.

Proof. Suppose $g \preceq_\varphi h$, $h \preceq_\psi g$. Since φ and ψ are rooted one has $\varphi = \psi^{-1}$, so φ, ψ are in fact rooted isomorphisms. Hence $g \equiv h$. $\square$

CONVENTION. We will usually identify equivalent graphs and replace $\equiv$ by $=$.

3.10. PROPOSITION. $\langle \mathrm{GRAPHS}, \preceq \rangle$ *is a partial ordering.*

PROOF. The relation $\preceq$ is clearly reflexive. Moreover $\preceq$ is antisymmetric by lemma 3.9 and transitive by lemma 3.6 (ii). $\square$

Now it is possible to indicate the relation between a graph and its complete unraveling.

3.11. LEMMA. *If g is coherent, then $g \preceq \mathrm{U}(g)$.*

PROOF. It is easily verified that $g \preceq_v \mathrm{U}(g)$, where

$$v(p) = n \qquad \text{if } p : r_g \rightsquigarrow n. \ \square$$

3.12. LEMMA. *Let g be a tree, and g' a coherent graph. Then*

$$g \preceq g' \ \Rightarrow \ g = g'.$$

PROOF. Suppose $g \preceq_\varphi g'$. One shows that $\varphi^{-1}(n)$ is a singleton for each $n \in g$, by induction on the length of the unique path $p : r_g \rightsquigarrow n$. Therefore φ is bijective. $\square$

3.13. LEMMA. (i) $g \preceq h \ \Rightarrow \ \mathrm{U}(g) \preceq \mathrm{U}(h)$.
(ii) $g \preceq h \ \Rightarrow \ \mathrm{U}(g) = \mathrm{U}(h)$.

PROOF. (i) Suppose $g \preceq_\varphi h$. Then $\mathrm{U}(g) \preceq \mathrm{U}(h)$ via the map $p \mapsto \varphi(p)$.
(ii) By (i) and lemmas 2.7 and 3.12. $\square$

We will now describe a general method to construct a 'quotient graph' from a graph combined with a certain equivalence relation on its nodes.

3.14. DEFINITION. (i) An *equivalence relation on g* is an equivalence relation on N_g.
(ii) Let $\sim$ be a equivalence relation on g. Then $\sim$ is *homogeneous* if for all $m, n \in g$ one has

$$\begin{array}{rcll} m \sim n & \Rightarrow & \mathit{symb}(m) = \mathit{symb}(n) & \& \\ & & \mathit{args}(m)_i \sim \mathit{args}(n)_i & \text{for all } i. \end{array}$$

3.15. DEFINITION. Let $\sim$ be a homogeneous equivalence relation on g. The *quotient graph* of g *modulo* $\sim$ (notation $g/_\sim$) is defined as follows. Set $g/_\sim = h$, where

$$\begin{array}{rcl} N_h & = & N_{g/_\sim} \quad (N_g \text{ modulo } \sim), \\ \mathit{symb}_h([m]) & = & \mathit{symb}_g(m), \\ \mathit{args}_h([m])_i & = & [\mathit{args}_g(m)_i], \\ r_h & = & [r_g]. \end{array}$$

Here $[m]$ denotes the equivalence class of m modulo $\sim$. Note that h is well-defined since $\sim$ is homogeneous.

3.16. DEFINITION. Let $\sim_1$ and $\sim_2$ be relations on a set A. We say that $\sim_1$ is a *refinement* of $\sim_2$ (notation $\sim_1 \subseteq \sim_2$) if for all $a, b \in A$

$$a \sim_1 b \;\Rightarrow\; a \sim_2 b.$$

(I.e. $\sim_1$ is included in $\sim_2$ if one considers relations as sets of ordered pairs.)

3.17. PROPOSITION. *Let $\sim_1$ and $\sim_2$ be homogeneous equivalence relations on g. Then*

$$\sim_1 \subseteq \sim_2 \;\Rightarrow\; g/_{\sim_2} \preceq g/_{\sim_1}.$$

PROOF. The map φ, defined by

$$\varphi([m]_{\sim_1}) = [m]_{\sim_2}$$

is a homomorphism from $g/_{\sim_1}$ to $g/_{\sim_2}$. $\square$

3.18. COROLLARY. *Let $\sim$ be homogeneous on g. Then*

$$g/_{\sim} \preceq g.$$

PROOF. By proposition 3.17 and the fact that $= \subseteq \sim$. $\square$

We now consider a special class of homogeneous relations.

3.19. DEFINITION. Let g, g' be such that $g' \preceq_\varphi g$. Define the relation $\sim_\varphi$ on N_g by

$$m \sim_\varphi n \;\Leftrightarrow\; \varphi(m) = \varphi(n).$$

Clearly $\sim_\varphi$ is an equivalence relation.

3.20. LEMMA. $g' \preceq_\varphi g \;\Rightarrow\; \sim_\varphi$ *is homogeneous on g.*

PROOF. Let $m, n \in g$. Suppose $m \sim_\varphi n$. Then

$$\begin{aligned} symb_g(m) &= symb_{g'}(\varphi(m)), \quad \text{since } \varphi \text{ is a homomorphism} \\ &= symb_{g'}(\varphi(n)) \\ &= symb_g(n). \end{aligned}$$

Moreover

$$\begin{aligned} \varphi(args_g(m)_i) &= args_{g'}(\varphi(m))_i \\ &= args_{g'}(\varphi(n))_i \\ &= \varphi(args_g(n)_i), \end{aligned}$$

which shows $args_g(m)_i \sim_\varphi args_g(n)_i$. $\square$

NOTATION. $g/_\varphi = g/_{\sim_\varphi}$.

3.21. LEMMA. (i) $g' \preceq_\varphi g \;\Rightarrow\; g/_\varphi \preceq g$.

(ii) $g' \preceq_\varphi g \;\Rightarrow\; g' \preceq g/_\varphi$.
(iii) $g' \preceq_\varphi g$, *g coherent* $\Rightarrow\; g' = g/_\varphi$.

PROOF. (i) By corollary 3.18.
(ii) One easily verifies that $g' \preceq_\psi g/_\varphi$, where

$$\psi([m]) = \varphi(m).$$

(iii) By coherence φ is surjective. Hence $g' \succeq_\psi g/_\varphi$, where

$$\psi(m') = \varphi^{-1}(m').$$

Now the equivalence follows from (ii). □

Below the following result will be proved. If two graphs have the same (possibly infinite) complete unraveling, then there exists a minimal common unraveling, which is finite if the graphs in question are finite.

3.22. DEFINITION. Two graphs g and g' are *tree equivalent* (notation $g \sim_{\mathrm{T}} g'$) if $\mathrm{U}(g) = \mathrm{U}(g')$.

3.23. PROPOSITION. *Let g, g' be finite coherent graphs. Suppose $g \sim_{\mathrm{T}} g'$. Then there exists a finite graph h such that*
(1) *$g \preceq h$ and $g' \preceq h$;*
(2) *if $g \preceq h_1$ and $g' \preceq h_1$, then $h \preceq h_1$.*

PROOF. Suppose $g \sim_{\mathrm{T}} g'$. Set $u = \mathrm{U}(g) = \mathrm{U}(g')$. Then by lemma 3.11 one has $g \preceq_\varphi u$ and $g' \preceq_{\varphi'} u$ for some φ, φ'. Now define a relation $\sim$ on u by

$$m \sim n \;\Leftrightarrow\; m \sim_\varphi n \;\&\; m \sim_{\varphi'} n$$

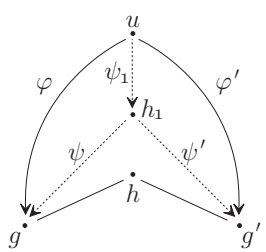
(i.e. $\sim \;=\; \sim_\varphi \cap \sim_{\varphi'}$ in set-theoretic terms). Note that $\sim$ is homogeneous on u, and $\sim \subseteq \sim_\varphi$, $\sim \subseteq \sim_{\varphi'}$. Define $h = u/_\sim$. Then h is finite since both $\sim_\varphi$ and $\sim_{\varphi'}$ give rise to a finite partition of u. Moreover

$$\begin{aligned} g &= u/_\varphi, && \text{by lemma 3.21 (iii)} \\ &\preceq u/_\sim, && \text{by propostion 3.17} \\ &= h. \end{aligned}$$

Similarly one shows $g' \preceq h$, thus establishing (1). As to (2), suppose $g \preceq_\psi h_1$ and $g' \preceq_{\psi'} h_1$. Then $h_1 = u/_{\psi_1}$ for some ψ_1.

Claim. $\sim_{\psi_1} \subseteq \sim$.

Proof. First note that $g \preceq_{\psi \circ \psi_1} u$ and $g' \preceq_{\psi' \circ \psi_1} u$.



Hence $\psi \circ \psi_1 = \varphi$ and $\psi' \circ \psi_1 = \varphi'$ by the uniqueness of rooted homomorphisms. Let $m, n \in u$. Then

$$\begin{aligned} m \sim_{\psi_1} n &\Rightarrow m \sim_{\psi\circ\psi_1} n,\ m \sim_{\psi'\circ\psi_1} n \\ &\Rightarrow m \sim_{\varphi} n,\ m \sim_{\varphi'} n, \quad \text{by the remark above} \\ &\Rightarrow m \sim n, \end{aligned}$$

which proves the claim. Now one has

$$\begin{aligned} h &= {}^{u}\!/_{\sim} \\ &\preceq {}^{u}\!/_{\psi_1} \\ &= h_1. \ \square \end{aligned}$$

NOTATION. The (unique) h mentioned in proposition 3.23 is called the *supremum* of g and g' and is denoted by $g \sqcup g'$.

3.24. DEFINITION. Let $g, g' \in \mathbb{G}$.

(i) g and g' are *upward compatible* (notation $g \uparrow g'$) if for some $h \in \mathbb{G}$ one has $g \preceq h$ and $g' \preceq h$.

(ii) g and g' are *downward compatible* (notation $g \downarrow g'$) if for some $h \in \mathbb{G}$ one has $h \preceq g$ and $h \preceq g'$.

3.25. THEOREM. $g \downarrow g' \ \Rightarrow \ g \uparrow g'$.

PROOF. Suppose $g \downarrow g'$. Then $g \sim_{\mathrm{T}} g'$. Hence by proposition 3.23 one has $g \uparrow g'$. $\square$

The method applied above can be used to obtain some domain theoretic results. Remember that $\mathbb{G}$ is the collection of finite graphs.

3.26. DEFINITION. The collection of *regular* graphs (notation $\mathbb{G}^*$) defined as follows.

$$\mathbb{G}^* = \{g \in \mathrm{GRAPHS} \mid g' \preceq g \ \text{ for some } g' \in \mathbb{G}\}.$$

Note that $\mathrm{U} : \mathbb{G} \to \mathbb{G}^*$.

3.27. DEFINITION. An *ascending chain* (in the structure $\langle \mathbb{G}^*, \preceq \rangle$) is a sequence $\gamma = g_0, g_1, g_2, \ldots$ of elements of $\mathbb{G}^*$ such that

$$g_0 \preceq g_1 \preceq g_2 \preceq \cdots .$$

The abbreviation $\gamma \preceq g$ stands for $\forall i\ g_i \preceq g$.

It will now be shown that $\langle \mathbb{G}^*, \preceq \rangle$ is an unpointed cpo, by a straightforward generalization of the proof of proposition 3.23.

3.28. PROPOSITION. *Every ascending chain has a least upper bound. That is, for each ascending chain γ there is a graph g such that*

(1) $\gamma \preceq g$,

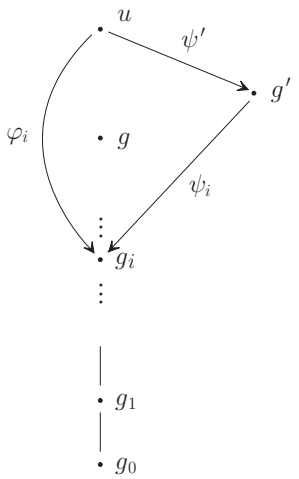
(2) *if $\gamma \preceq g'$ then $g \preceq g'$, for any $g' \in \mathbb{G}$.*

PROOF. Let $\gamma = g_0, g_1, \ldots$ be an ascending chain. Set $u = \mathrm{U}(g_0)$. Then by lemma 3.13 one has $\mathrm{U}(g_i) = u$ for each i. Therefore $\gamma \preceq u$, say $g_i \preceq_{\varphi_i} u$. Now define $\sim$ on u by

$$m \sim n \;\Leftrightarrow\; \forall i \;\; m \sim_{\varphi_i} n.$$

Set $g = {}^{u}\!/_{\sim}$. Then $\sim$ is homogeneous, and $\sim \subseteq \sim_{\varphi_i}$ for all i. Hence $\gamma \preceq g$. Moreover suppose $\gamma \preceq g'$, say via ψ_i ($i \in \mathbb{N}$). Observe that $\mathrm{U}(g') = u$, say $g' \preceq_{\psi'} u$.

Claim. $\sim_{\psi'} \subseteq \sim$.



Proof. Note that for all i one has $\varphi_i = \psi_i \circ \psi'$, by the uniqueness of rooted homomorphisms. Then for each $i \in \mathbb{N}$

$$\begin{aligned} m \sim_{\psi'} n \quad &\Rightarrow \quad m \sim_{\psi_i \circ \psi'} n \\ &\Rightarrow \quad m \sim_{\varphi_i} n. \end{aligned}$$

Hence $m \sim n$, which proves the claim. Now one has

$$\begin{aligned} g \;&=\; {}^{u}\!/_{\sim} \\ &\preceq\; {}^{u}\!/_{\sim_{\psi'}} \\ &=\; g'. \;\square \end{aligned}$$

NOTATION. The unique g above is called the *supremum* of γ, notation $g = \sqcup \gamma$.

Next we consider descending chains.

3.29. DEFINITION. (i) A *descending chain* is a sequence $\gamma = g_0, g_1, g_2, \ldots$ such that

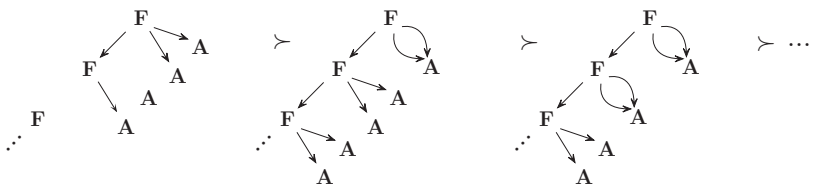
$$g_0 \succeq g_1 \succeq g_2 \succeq \cdots.$$

(ii) γ is *strictly descending* if

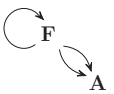
$$g_0 \succ g_1 \succ g_2 \succ \cdots.$$

3.30. PROPOSITION. (i) *$\langle \mathbb{G}^*, \preceq \rangle$ is not wellfounded.*
(ii) *$\langle \mathbb{G}, \preceq \rangle$ is wellfounded.*

PROOF. (i) One can construct an infinite strictly descending chain, as suggested by the following figure.



Note that all graphs in this chain are unravelings of



so indeed the above chain is in $\mathbb{G}^*$.

(ii) If $g \succ g'$ and g, g' are finite, then the number of nodes in g' is strictly smaller than the number of nodes in g. □

3.31. PROPOSITION. *Each descending chain has a greatest lower bound.*

PROOF. Let $\gamma = g_0, g_1, g_2, \ldots$ be a descending chain, say

$$g_0 \succeq_{\varphi_0} g_1 \succeq_{\varphi_1} g_2 \succeq_{\varphi_2} \cdots.$$

Define $\sim_i$ on g_0 by

$$\sim_i \; = \; \sim_{\varphi_i \circ \varphi_{i-1} \circ \cdots \circ \varphi_0},$$

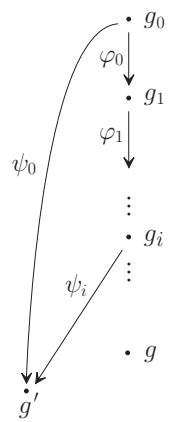
and $\sim$ by

$$m \sim n \;\Leftrightarrow\; \exists i \;\; m \sim_i n.$$

Then $\sim$ is homogeneous since $\sim_i \subseteq \sim_{i+1}$ for each i. Set $g = {}^{g_0}\!/_{\sim}$. Let $i \in \mathbb{N}$. Then

$$\begin{array}{rcl} g & = & {}^{g_0}\!/_{\sim} \\ & \preceq & {}^{g_0}\!/_{\sim_i}, \quad \text{since } \sim_i \subseteq \sim \\ & = & g_i. \end{array}$$

Therefore $g \preceq \gamma$. Now suppose $g' \preceq \gamma$, say $g' = {}^{g_i}\!/_{\psi_i}$.



Claim. $\sim \subseteq \sim_{\psi_0}$.

Proof. Suppose $m \sim n$. Determine i such that $m \sim_i n$, i.e. $m \sim_{\varphi_i \circ \cdots \circ \varphi_0} n$. Then also $m \sim_{\psi_i \circ \varphi_i \circ \cdots \circ \varphi_0} n$. But $\psi_i \circ \varphi_i \circ \cdots \circ \varphi_0 = \psi_0$, so $m \sim_{\psi_0} n$, which proves the claim. Hence

$$\begin{aligned} g' &= g_0/_{\sim_{\psi_0}} \\ &\preceq g_0/_{\sim} \\ &= g. \ \square \end{aligned}$$

NOTATION. This construction yields the so called *infimum* of γ, denoted as $\sqcap\gamma$.

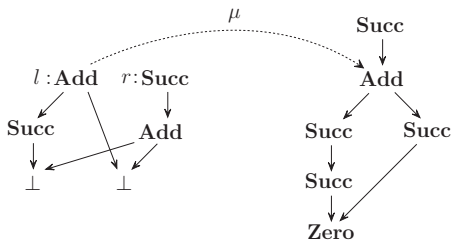
3.32. REMARK. Note that the constructions in 3.28 and 3.31 make use of a common upper bound (u and g_0 respectively) of the chains in question. It is possible to generalize the results about least upper bounds and greatest lower bounds to arbitrary chains, i.e. to sets $G \subseteq \mathbb{G}^*$ such that

$$\forall g, g' \in G \qquad g \preceq g' \text{ or } g \succeq g',$$

by taking the complete unraveling of the elements of G as such a common upper bound.

4. Graph rewriting

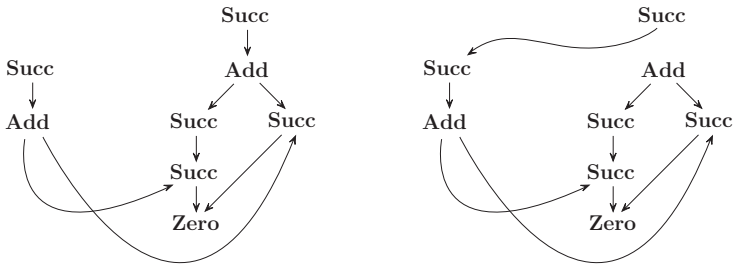
Rewrite rules specify transformations of graphs. Each rewrite rule is represented by a special graph containing two roots. These roots determine the left-hand side (the *pattern*) and the right-hand side of the rule. Variables are represented by special 'empty nodes'. Let R be some rewrite rule. A graph g can be *rewritten* according to R if R is applicable to g, i.e. the pattern of R *matches* g. A *match* is a mapping from the pattern of R to a subgraph of g that preserves the node structure. The following picture gives the graph representation of the rule $\mathbf{Add}(\mathbf{Succ}(x), y) \to \mathbf{Succ}(\mathbf{Add}(x, y))$ and a match μ.



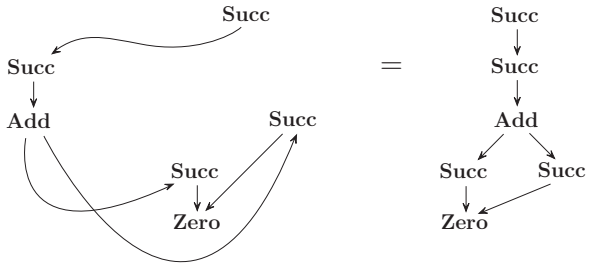
Instead of using names for variables in rewrite rules, multiple occurrence of the same variable is indicated via sharing. E.g. all occurrences of the variable x in the above rule are represented by references to the same empty node.

The combination of a rule and a match is called a *redex*. If a redex has been determined, the graph can be rewritten according to the structure of the right-hand side of the rule involved. This is done in three steps. Firstly, the graph is *extended* with an instance of the right-hand side of the rule. The connections from the new part

with the original graph are determined by μ. Then all references to the root of the redex are *redirected* to the root of the right-hand side.



Finally all unreachable nodes are removed by performing *garbage collection*.



This procedure is formalized in the rest of the present section.

4.1. Definition. Let $\perp$ be a special symbol in Σ with arity 0. Let g be a graph.

(i) The set of *empty nodes* of g (notation g°) is the collection

$$g^\circ = \{n \in g \mid symb_g(n) = \perp\}.$$

(ii) The set of *non-empty nodes* (or the *interior*) of g is denoted by $g^\bullet$. So $N_g = g^\circ \cup g^\bullet$.

(iii) g is *closed* if $g^\circ = \emptyset$.

The objects on which computations are performed are closed graphs; the others are used as auxiliary objects, e.g. for defining graph rewrite rules.

4.2. Definition. A *term graph* is a closed rooted graph.

4.3. Definition. (i) A *term graph rewrite rule* (or *rule* for short) is a triple

$$R = \langle g, l, r \rangle$$

where g is a (possibly open) graph, and $l, r \in g$ (called the *left root* and *right root* of R), such that

(1) $(g \mid l)^\bullet \neq \emptyset$;

(2) $(g \mid r)^\circ \subseteq (g \mid l)^\circ$.

(ii) R is a *selector rule* if $(g \mid r)^\bullet = \emptyset$.

(iii) If $symb_g(l) = \mathbf{F}$ then R is said to be a *rule for* $\mathbf{F}$.

Here condition (1) expresses that the left-hand side of the rewrite rule should not be just a variable. Moreover condition (2) states that all variables occurring on the right-hand side of the rule should also occur on the left-hand side.

NOTATION. We will write $R \mid l$, $R \mid r$ for $g_R \mid l_R$, $g_R \mid r_R$ respectively.

4.4. DEFINITION. Let p, g be graphs.

(i) A *matching homomorphism* (or *match* for short) is a function $\mu : N_p \to N_g$ such that for all $n \in p^\bullet$

$$\begin{aligned} symb_p(n) &= symb_g(\mu(n)), \\ \mu(args_p(n)_i) &= args_g(\mu(n))_i. \end{aligned}$$

In this case we write $\mu : p \underset{\mathrm{m}}{\rightrightarrows} g$.

(ii) μ is a *rooted match* from p to g (notation $\mu : p \underset{\mathrm{rm}}{\rightrightarrows} g$) if $\mu : p \underset{\mathrm{m}}{\rightrightarrows} g$ and $\mu(r_p) = \mu(r_g)$.

(iii) By $p \underset{\mathrm{m}}{\rightarrow} g$ we will denote that there exists a match $\mu : p \underset{\mathrm{m}}{\rightarrow} g$. Analogously we write $p \underset{\mathrm{rm}}{\rightarrow} g$ for rooted matches.

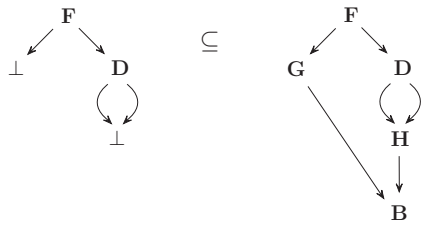
4.5. DEFINITION. Let $s, g \in \mathbb{G}$. Then s is a *segment* of g (notation $s \subseteq g$) if s is coherent, and s results (up to isomorphism) from g by replacing some subgraphs by $\bot$. More formally, $s \subseteq g$ if there exists an injective match

$$\iota : s \underset{\mathrm{m}}{\rightrightarrows} g.$$

Observe that such an injection is unique; it will be referred to as $\iota_{s,g}$, or just ι if there is no danger of confusion.

Intuitively, $s \subseteq g$ if s is an initial ('rooted') part of g.

4.6. EXAMPLE.

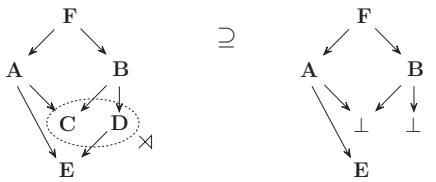


The selection of a segment of a given graph is indicated as follows. The construction itself is left implicit.

4.7. DEFINITION. Let g be a graph and $S \subseteq N_g$. The *segment* of g *up to* S (notation $g \rtimes S$) is obtained by replacing the nodes in S by $\bot$.

A segment is obtained by cutting off subgraphs: the symbol $\rtimes$ is supposed to resemble a pair of scissors.

4.8. Example.



The correspondence between matching homomorphisms and ordinary homomorphisms is expressed by the following straightforward result.

4.9. Lemma. $p \underset{\text{rm}}{\rightrightarrows} g \;\Leftrightarrow\; p \succeq q$ *for some segment* $q \subseteq g$.

Proof. ($\Rightarrow$) Suppose $\mu : p \underset{\text{rm}}{\rightrightarrows} g$. It can easily be verified that

$$p \succeq g \rtimes \mu(p^\bullet).$$

($\Leftarrow$) Suppose $q \subseteq g$, and $p \succeq_\varphi q$. Then

$$\iota_{q,g} \circ \varphi : p \underset{\text{rm}}{\rightrightarrows} g. \;\square$$

Traditionally, a redirection in a graph g is an operation which replaces all references to a given node by references to another one. This notion is generalized in the following definition.

4.10. Definition. (i) A *redirection map* (or simply *redirection*) on g is a function $\rho : N_g \to N_g$.

(ii) ρ is *projective* if $\rho \circ \rho = \rho$.

(iii) Let $N \subseteq N_g$. N is *ρ-closed* if $\rho(N) \subseteq N$.

(iv) Let h be a (sub)graph. Then ρ is *applicable* to h if N_h is ρ-closed.

A redirection map on g induces a transformation of g, as indicated before.

4.11. Definition. Let ρ be a redirection on g. The result of *applying* ρ to g (notation $\rho(g)$) is the graph

$$\rho(g) = \langle N_g, \mathit{symb}_g, \overline{\rho} \circ \mathit{args}_g, \rho(r_g) \rangle$$

where $\overline{\rho}$ is the canonical extension of ρ to finite sequences, i.e.

$$\overline{\rho}(n_1, \ldots, n_k) = (\rho(n_1), \ldots, \rho(n_k)).$$

4.12. Definition. (i) Let ρ be a redirection on g. The *field* of ρ (notation $\mathrm{Fld}(\rho)$) is the set

$$\mathrm{Fld}(\rho) = \{n \in N_g \mid \rho(n) \neq n\}.$$

(ii) ρ is a *single redirection* if $\mathrm{Fld}(\rho)$ is a singleton. In this case ρ is written as $[m \mapsto n]$ if $\mathrm{Fld}(\rho) = \{m\}$ and $\rho(m) = n$. The application $[m \mapsto n](g)$ is traditionally denoted as $g[m := n]$.

4.13. Remark. Note that $(\rho_1 \circ \rho_2)(g) = \rho_1(\rho_2(g))$ only holds if ρ_2 is applicable to g.

In some cases a redirection can be written as a sequence of single redirections.

4.14. Lemma. *Let ρ be a redirection such that ρ is projective and* $\mathrm{Fld}(\rho)$ *is finite, say* $\mathrm{Fld}(\rho) = \{m_1, \ldots, m_k\}$. *Then*

$$\rho = [m_1 \mapsto \rho(m_1)] \circ \cdots \circ [m_k \mapsto \rho(m_k)].$$

Proof. Trivial. □

Notation. For ρ as in the previous lemma we write

$$\rho = [m_1 \mapsto \rho(m_1), \ldots, m_k \mapsto \rho(m_k)]$$

and

$$\rho(g) = g[m_1 := \rho(m_1), \ldots, m_k := \rho(m_k)]$$

respectively. Note that this is a sound denotation since the result is independent of the order in which the single redirections are carried out.

4.15. Definition. Two redirections ρ_1 and ρ_2 are *independent* if $\rho_1 \circ \rho_2 = \rho_2 \circ \rho_1$.

4.16. Remark. If ρ_1, ρ_2 are projective and ρ_1, ρ_2 are independent, then $\rho_1 \circ \rho_2$ is also projective.

The effect of a single redirection on paths is indicated in the following, trivial but useful remark.

4.17. Remark. Let g be a graph.

(i) Let $n, m \in g$. Let p be a path in g such that $n \notin p$. Then p is a path in $g[n := m]$.

(ii) If $p_1 = (n_0, i_0, \ldots, n_{k-1}, i_{k-1}, n_k)$ and $p_2 = (m_0, j_0, \ldots, m_l)$ are paths in g, and

$$\begin{aligned} &\forall 0 \leq t < k \quad n_t \neq n_k, \\ &\forall 0 < t \leq l \quad m_t \neq n_k, \end{aligned}$$

then $(n_0, i_0, \ldots, n_{k-1}, i_{k-1}, m_0, j_0, \ldots, m_l)$ is a path in $g[n_k := m_0]$.

Below a general method for extending graphs will be described.

4.18. Definition. Let N, A be sets of nodes. The *disjoint set extension* of N with A (notation $N \cup^+ A$) is the set $N \cup A^*$, where $A^* = \{a^* \mid a \in A\}$ is a set of fresh nodes associated with A.

4.19. Remark. There is a canonical way of extending an operation f on N, A to an operation f^* on $N \cup^+ A$. If no confusion can arise, we keep writing f, a for f^*, a^* respectively. The same will be done for subsets of N, A and the corresponding subsets of $N \cup^+ A$.

4.20. Definition. Let g be a graph.

(i) Let h be a graph, $H \subseteq N_h$, and $\mu : (N_h - H) \to N_g$. The *extension* of g with $\langle h, H\rangle$ according to μ (notation $g +_\mu \langle h, H\rangle$) is defined as follows. Set $g +_\mu \langle h, H\rangle = k$, where

$$\begin{array}{rcll} N_k & = & N_g \cup^+ H; & \\ symb_k(n) & = & symb_g(n) & \text{if } n \in g, \\ & = & symb_h(n) & \text{otherwise;} \\ args_k(n)_i & = & args_g(n)_i & \text{if } n \in g, \\ & = & args_h(n)_i & \text{if } n, args_h(n) \in H, \\ & = & \mu(args_h(n)_i) & \text{otherwise;} \\ r_k & = & r_g. & \end{array}$$

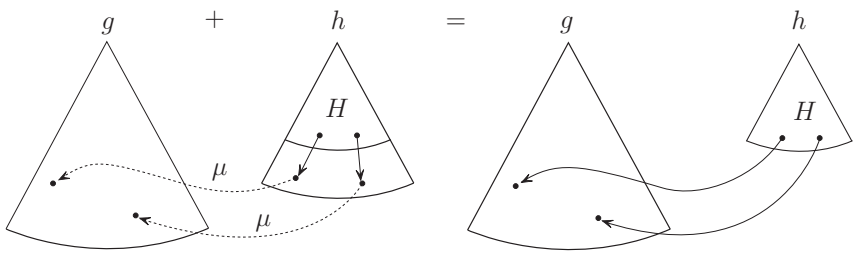
(ii) An *extension specification* (or *extension* for short) for g is a triple $E = \langle h, H, \mu\rangle$ with h, H and μ as above. Then the extension of g with E is

$$g + E = g +_\mu \langle h, H\rangle.$$

(iii) The *extension root* of E (notation $\mathrm{r}(E)$) is defined as follows.

$$\begin{array}{rcll} \mathrm{r}(E) & = & r_h & \text{if } r_h \in H, \\ & = & \mu(r_h) & \text{otherwise.} \end{array}$$

This construction can be visualized as follows.



NOTATION. We use $\langle h, \mu\rangle$ as an abbreviation for $\langle h, h^\bullet, \mu\rangle$. Analogously we write $g +_\mu h$ for $g +_\mu \langle h, h^\bullet\rangle$.

4.21. REMARK. Sometimes we will loosely consider $\langle h, H, \mu\rangle$ as an extension, even if $H \not\subseteq N_h$. This is to be understood as $\langle h, H \cap N_h, \mu\rangle$.

4.22. DEFINITION. Let g be a rooted graph. The result of performing *garbage collection* on g (notation $\mathrm{GC}(g)$) is the graph obtained from g by deleting all nodes that are not reachable from its root, i.e.

$$\mathrm{GC}(g) = g \mid r_g.$$

The three basic operations extension, redirection, and garbage collection are combined into a single operation called *replacement*. This mechanism corresponds to *substitution* in lambda calculus and term rewriting.

4.23. Definition. Let E be an extension of g, and let $n \in g$. The result of *replacing* n by E in g (notation $g[n := E]$) is defined by

$$g[n := E] = \mathrm{GC}((g + E)[n := \mathrm{r}(E)]).$$

Following Barendregt et al. (1987b), one can distinguish the following steps in the construction of $g[n := E]$.

$$\begin{array}{lll} g & \mapsto & g + E \qquad\qquad\qquad\quad \text{(build phase)}, \\ & \mapsto & (g + E)[n := \mathrm{r}(E)] \quad \text{(redirection phase)}, \\ & \mapsto & \mathrm{GC}((g + E)[n := \mathrm{r}(E)]) \quad \text{(garbage collection phase)}. \end{array}$$

4.24. Definition. Let g be a graph, and $\mathcal{R}$ a set of rewrite rules.

(i) An $\mathcal{R}$-*redex* in g (or just *redex*) is a tuple

$$\Delta = \langle R, \mu \rangle$$

where $R \in \mathcal{R}$, and $\mu : (R \mid l) \underset{\mathrm{m}}{\to} g$.

(ii) The extension associated with Δ is

$$E_\Delta = \langle R \mid r,\ (R \mid r)^\bullet - (R \mid l)^\bullet,\ \mu \rangle.$$

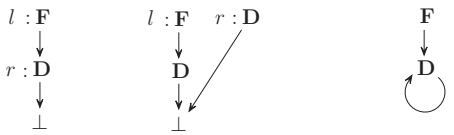
We write $g + \Delta$ instead of $g + E_\Delta$, and $g[n := \Delta]$ instead of $g[n := E_\Delta]$. Moreover $\mathrm{r}(\Delta)$ means $\mathrm{r}(E_\Delta)$.

(iii) Let $\Delta = \langle R, \mu \rangle$ be a redex. The result g' of *contracting* Δ in g is given by the replacement

$$g' = g[\mu(l) := \Delta].$$

In this case we say that g reduces to g' via Δ (notation $g \overset{\Delta}{\underset{\mathcal{R}}{\to}} g'$, or just $g \underset{\mathcal{R}}{\to} g'$).

The definition of E_Δ allows nonempty nodes in the left-hand side of a rule to be reused in the right-hand side. Note the difference between the following rewrite rules, when applied to the given graph.



The following technical results concern the interaction between the three basic operations.

4.25. Lemma. *Let E_1, E_2 be extensions of g. Then*

$$g + E_1 + E_2 = g + E_2 + E_1.$$

Proof. Trivial. □

4.26. Lemma. (i) $\mathrm{GC}(\mathrm{GC}(g)) = \mathrm{GC}(g)$.

(ii) $\mathrm{GC}(g + E) = \mathrm{GC}(g)$.

PROOF. Trivial. □

4.27. LEMMA. *Let p, g be graphs, and let $\mu : p \xrightarrow[\mathrm{m}]{} g$ be a match. Then for any redirection ρ of g*

$$\mu(p^\bullet) \cap \mathrm{Fld}(\rho) = \emptyset \;\Rightarrow\; \rho \circ \mu : p \xrightarrow[\mathrm{m}]{} g.$$

PROOF. Note that $\rho \circ \mu(m)$ only differs from $\rho(m)$ if $m \notin \mu(p^\bullet)$. Thus like μ also $\rho \circ \mu : p \xrightarrow[\mathrm{m}]{} g$. □

4.28. LEMMA. *Let g be a graph, $E = \langle h, H, \mu \rangle$ an extension of g and $n \in g$. Furthermore, let ρ be a redirection. (Below $\restriction$ denotes restriction.)*

(i) *If ρ is applicable to $g \mid n$ then*

$$\rho(g \mid n) \mid \rho(n) = \rho(g) \mid \rho(n).$$

(ii) *Let ρ' be a redirection such that ρ, ρ' are applicable to $g \mid n$. Then*

$$\rho \restriction (g \mid n) = \rho' \restriction (g \mid n) \;\Rightarrow\; \rho(g) \mid \rho(n) = \rho'(g) \mid \rho'(n).$$

(iii) *If ρ is applicable to $(g \mid n) + E$ then*

$$\rho((g \mid n) + E) \mid \rho(n) = \rho(g + E) \mid \rho(n).$$

(iv) *If ρ is applicable to $g +_\mu \langle \mathrm{GC}(h), H \rangle$ then*

$$\rho(g +_\mu \langle \mathrm{GC}(h), I \rangle) \mid \rho(n) = \rho(g +_\mu \langle h, I \rangle) \mid \rho(n).$$

PROOF. (i) For all nodes $m \in g \mid n$ one obviously has $\rho(g \mid n) \mid m = \rho(g) \mid m$. From the assumption that ρ is applicable to $g \mid n$ it follows that $\rho(n) \in g \mid n$ and hence the equality holds.
(ii)-(iv) can be proved similarly. □

By taking the root of the graph as n in the above lemma the following properties concerning garbage collection are immediately obtained.

4.29. PROPOSITION. *Let g be a graph, $E = \langle h, I, \mu \rangle$ an extension of g, and let ρ be a redirection.*

(i) *If ρ is applicable to $\mathrm{GC}(g)$ then*

$$\mathrm{GC}(\rho(\mathrm{GC}(g))) = \mathrm{GC}(\rho(g)).$$

(ii) *Let ρ' be a redirection such that ρ, ρ' are applicable to $\mathrm{GC}(g)$. Then*

$$\rho \restriction \mathrm{GC}(g) = \rho' \restriction \mathrm{GC}(g) \;\Rightarrow\; \mathrm{GC}(\rho(g)) = \mathrm{GC}(\rho'(g)).$$

(iii) *If ρ is applicable to $\mathrm{GC}(g) + E$ then*

$$\mathrm{GC}(\rho(\mathrm{GC}(g) + E)) = \mathrm{GC}(\rho(g) + E).$$

(iv) *If ρ is applicable to $g +_\mu \langle \mathrm{GC}(h), I\rangle$ then*

$$\mathrm{GC}(\rho(g +_\mu \langle \mathrm{GC}(h), I\rangle)) = \mathrm{GC}(\rho(g +_\mu \langle h, I\rangle)).$$

4.30. PROPOSITION. *Let $\langle h, H, \mu\rangle$ be an extension of g, and let ρ_1, ρ_2 be redirections of g, h respectively such that $N_h - H$ is ρ_2-closed. Then*

$$\rho_1 \circ \mu \circ \rho_2(g +_{\mu\circ\rho_2} \langle h, H\rangle) = \rho_1(g) +_{\rho_1\circ\mu} \langle \rho_2(h), H\rangle.$$

(Here μ is tacitly extended to all nodes of the extended graph, by defining $\mu(n) = n$ if $n \notin N_h - H$. The same is done for ρ_1 and ρ_2.)

PROOF. Set $g_1 = \rho_1 \circ \mu \circ \rho_2(g +_{\mu\circ\rho_2} \langle h, H\rangle)$ and $g_2 = \rho_1(g) +_{\rho_1\circ\mu} \langle \rho_2(h), H\rangle$. Then $g_1 = g_2$, as is shown by the following case distinction.

Case 1. $n \in g$. Then

$$\begin{aligned} args_1(n)_i &= \rho_1 \circ \mu \circ \rho_2(args_g(n)_i) \\ &= \rho_1(args_g(n)_i), \quad \text{since } \mu \circ \rho_2(args_g(n)_i) = args_g(n)_i \\ &= args_2(n)_i. \end{aligned}$$

Case 2. $n, \rho_2(args_h(n)_i) \in H$. Since $N_h - H$ is ρ_2-closed it follows that also $args_h(n)_i \in H$. Then

$$\begin{aligned} args_1(n)_i &= \rho_1 \circ \mu \circ \rho_2(args_h(n)_i) \\ &= \rho_2(args_h(n)_i), \quad \text{since } \rho_1 \circ \mu(m) = m \text{ for } m \in H \\ &= args_2(n)_i. \end{aligned}$$

Case 3. $n, args_h(n)_i \in H$ and $\rho_2(args_h(n)_i) \in N_h - H$. Then

$$\begin{aligned} args_1(n)_i &= \rho_1 \circ \mu \circ \rho_2(args_h(n)_i) \\ &= args_2(n)_i. \end{aligned}$$

Case 4. $n \in H$ and $args_h(n)_i, \rho_2(args_h(n)_i) \in N_h - H$. Using $\mu \circ \rho_2(\mu \circ \rho_2(m)) = \mu \circ \rho_2(m)$ for $m \in N_h - H$ one derives

$$\begin{aligned} args_1(n)_i &= \rho_1 \circ \mu \circ \rho_2(\mu \circ \rho_2(args_h(n)_i)) \\ &= \rho_1 \circ \mu \circ \rho_2(args_h(n)_i) \\ &= args_2(n)_i. \ \square \end{aligned}$$

5. Graph rewrite systems

A collection of graphs and a set of rewrite rules can be combined into a graph rewrite system. The special class of so called orthogonal graph rewrite systems will be studied in more detail.

5.1. DEFINITION. A *term graph rewrite system* (TGRS) is a tuple

$$\mathcal{T} = \langle \mathcal{G}, \mathcal{R}\rangle$$

where $\mathcal{R}$ is a set of rewrite rules, and $\mathcal{G} \subseteq \mathbb{G}$ is a set of term graphs which is closed under $\mathcal{R}$-reduction.

5.2. Definition. Let $\mathcal{T} = \langle \mathcal{G}, \mathcal{R} \rangle$ be a TGRS.
(i) A rule $R \in \mathcal{R}$ is *left-linear* if $R \mid l$ is a tree.
(ii) $\mathcal{T}$ is *left-linear* if each $R \in \mathcal{R}$ is left-linear.

It appears that under certain circumstances the order of reduction can be changed without influencing the final result.

The following definition distinguishes redexes on the basis of their relative position in a graph.

5.3. Definition. Let $\Delta_1 = \langle R_1, \mu_1 \rangle$ and $\Delta_2 = \langle R_2, \mu_2 \rangle$ be redexes in g.
(i) Δ_1 and Δ_2 are *disjoint* if

$$\mu_1(l_1) \notin \mu_2(R_2 \mid l_2)^\bullet, \qquad \mu_2(l_2) \notin \mu_1(R_1 \mid l_1)^\bullet.$$

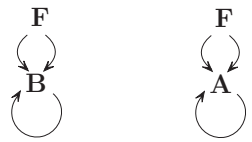
(ii) Δ_1 and Δ_2 *interfere* if

$$\mu_1(l_1) = \mathrm{r}(\Delta_2), \qquad \mu_2(l_2) = \mathrm{r}(\Delta_1).$$

If a graph contains redexes that are not disjoint, it is conceivable that different reduction orders may yield different results.

This is also the case if a graph contains interfering redexes, as is illustrated by the following graph and rewrite rules for **A** and **B**.

The respective results after contracting the **A**-redex and the **B**-redex are



5.4. Definition. Let $\mathcal{T} = \langle \mathcal{G}, \mathcal{R} \rangle$ be a TGRS.
(i) $\mathcal{T}$ is *regular* if for each $g \in \mathcal{G}$ the $\mathcal{R}$-redexes in g are pairwise disjoint.
(ii) $\mathcal{T}$ is *orthogonal* if $\mathcal{T}$ is both left-linear and regular.
(iii) $\mathcal{T}$ is *interference-free* if $\mathcal{G}$ does not contain any graph with interfering redexes.

It will be shown that for orthogonal interference-free TGRS's the reduction order can be changed. Some technical properties are proved first.

5.5. Lemma. *Let $n_1, n_2, m_1, m_2 \in \mathcal{N}$ with $n_1 \neq n_2$. If $n_1 \neq m_2$ or $n_2 \neq m_1$ then*

$$[n_1 \mapsto [n_2 \mapsto m_2](m_1)] \circ [n_2 \mapsto m_2] = [n_2 \mapsto [n_1 \mapsto m_1](m_2)] \circ [n_1 \mapsto m_1].$$

Proof. A matter of easy casuistics. □

5.6. Lemma. *Let g be a graph and $n_1, n_2 \in g$ with $n_1 \neq n_2$. Furthermore, let $E_1 = \langle h_1, H_1, \mu_1 \rangle$ and $E_2 = \langle h_2, H_2, \mu_2 \rangle$ be extensions of g. Write*

$$E_1' = \langle h_1, H_1, [n_2 \mapsto \mathrm{r}(E_2)] \circ \mu_1 \rangle, \qquad E_2' = \langle h_2, H_2, [n_1 \mapsto \mathrm{r}(E_1)] \circ \mu_2 \rangle.$$

If $n_1 \neq \mathrm{r}(E_2)$ or $n_2 \neq \mathrm{r}(E_1)$ then

$$g[n_1 := E_1][n_2 := E_2'] = g[n_2 := E_2][n_1 := E_1'].$$

Proof.

$$\begin{aligned}
& g[n_1 := E_1][n_2 := E_2'] \\
&\quad = \mathrm{GC}(\ (\mathrm{GC}((g + E_1)[n_1 := \mathrm{r}(E_1)]) + E_2')[n_2 := \mathrm{r}(E_2')]\ ) \\
&\quad = \mathrm{GC}(\ ((g + E_1)[n_1 := \mathrm{r}(E_1)] + E_2')[n_2 := \mathrm{r}(E_2')]\ ), \\
&\qquad\qquad \text{by proposition 4.29 (iii, i)} \\
&\quad = \mathrm{GC}(\ (g + E_1 + E_2)[n_1 := \mathrm{r}(E_1)][n_2 := \mathrm{r}(E_2')]\ ), \\
&\qquad\qquad \text{by proposition 4.30} \\
&\quad = \mathrm{GC}(\ (g + E_2 + E_1)[n_2 := \mathrm{r}(E_2)][n_1 := \mathrm{r}(E_1')]\ ), \\
&\qquad\qquad \text{by lemmas 4.25 and 5.5} \\
&\quad = \mathrm{GC}(\ ((g + E_2)[n_2 := \mathrm{r}(E_2)] + E_1')[n_1 := \mathrm{r}(E_1')]\ ) \\
&\quad = \mathrm{GC}(\ (\mathrm{GC}((g + E_2)[n_2 := \mathrm{r}(E_2)] + E_1')[n_1 := \mathrm{r}(E_1')]\ ) \\
&\quad = g[n_2 := E_2][n_1 := E_1'].\ \square
\end{aligned}$$

5.7. Lemma. *Let $\Delta_1 = \langle R_1, \mu_1 \rangle$ and $\Delta_2 = \langle R_2, \mu_2 \rangle$ be disjoint redexes in g. Suppose $g \overset{\Delta_1}{\underset{\mathcal{R}}{\to}} g'$. If $\mu_2(l_2) \in g'$, then*

$$\Delta_2' = \langle R_2, [\mu_1(l_1) \mapsto \mathrm{r}(\Delta_1)] \circ \mu_2 \rangle$$

is a redex in g'. Moreover $[\mu_1(l_1) \mapsto \mathrm{r}(\Delta_1)] \circ \mu_2(l_2) = \mu_2(l_2)$.

Proof. Note that $g' = \mathrm{GC}((g + \Delta)[\mu_1(l_1) := \mathrm{r}(\Delta_1)])$. By lemma 4.27 one has

$$[\mu_1(l_1) \mapsto \mathrm{r}(\Delta_1)] \circ \mu_2 : (R_1 \mid l_1)^\bullet \underset{\mathrm{m}}{\to} (g + \Delta)[\mu_1(l_1) := \mathrm{r}(\Delta_1)]$$

since Δ_1 and Δ_2 are disjoint. During garbage collection only unreachable nodes are removed; the graph structure is not affected. Hence if $\mu_2(l_2) \in g'$ then indeed $[\mu_1(l_1) \mapsto \mathrm{r}(\Delta_1)] \circ \mu_2 : (R_1 \mid l_1)^\bullet \underset{\mathrm{m}}{\to} g'$. The second property holds since $\mu_1(l_1) \neq \mu_2(l_2)$. $\square$

5.8. Definition. Let $g, g', \Delta_1, \Delta_2, \Delta_2'$ be as in the previous lemma.

(i) If $\mu_1(l_2) \in g'$, then Δ_2' is called the *residual* of Δ_2 in g' (relative to the reduction $g \overset{\Delta_1}{\underset{\mathcal{R}}{\to}} g'$), notation $\Delta_2 /\!\!/ \Delta_1$. If $\mu_1(l_2) \notin g'$ then Δ_2 has no residual in g'.

(ii) The reduction (consisting of zero or one step) contracting $\Delta_2 /\!\!/ \Delta_1$ in g' (if it exists) is denoted by $\overset{\Delta_2 /\!\!/ \Delta_1}{\underset{\mathcal{R}}{\to}}$.

5.9. Proposition. *Let $\mathcal{T} = \langle \mathcal{G}, \mathcal{R} \rangle$ be othogonal and interference-free. Suppose $g \overset{\Delta_1}{\underset{\mathcal{R}}{\to}} g_1$ and $g \overset{\Delta_2}{\underset{\mathcal{R}}{\to}} g_2$. Say $g_1 \overset{\Delta_2 /\!\!/ \Delta_1}{\underset{\mathcal{R}}{\to}} h_1$ and $g_2 \overset{\Delta_1 /\!\!/ \Delta_2}{\underset{\mathcal{R}}{\to}} h_2$. Then $h_1 = h_2$.*

PROOF. By lemma 5.7 and proposition 5.6. Note that $\mu_1(l_1) \neq \mathrm{r}(\Delta_2)$ or $\mu_1(l_2) \neq \mathrm{r}(\Delta_2)$ since Δ_1 and Δ_2 do not mutually interfere. $\square$

5.10. REMARK. If an orthogonal TGRS $\mathcal{T}$ is cycle-free, i.e. the graphs of $\mathcal{T}$ do not contain cycles, then $\mathcal{T}$ is interference-free.

ASSUMPTION. From now on we assume that all TGRS's in question are both orthogonal and cycle-free.

6. Combining rewriting and unraveling

Unraveling a graph might influence the redexes of that graph. The main result of this section states that unraveling and reduction can be interchanged.

First we describe how unraveling interacts with the redirection and extension operations.

6.1. LEMMA. *Let $g \succeq_\varphi h$, and let $n \in h$. Suppose h is cycle-free. Then the elements of $\varphi^{-1}(n)$ are 'mutually independent', i.e. for all $m_1, m_2 \in \varphi^{-1}(n)$ with $m_1 \neq m_2$*

$$m_1 \not\rightsquigarrow m_2 \quad \textit{and} \quad m_2 \not\rightsquigarrow m_1.$$

PROOF. Let $m_1, m_2 \in \varphi^{-1}(n)$. Suppose $m_1 \neq m_2$, and $p : m_1 \rightsquigarrow m_2$. Note that $|p| \geq 1$. Then $\varphi(p) : n \rightsquigarrow n$ and $|\varphi(p)| \geq 1$, contradicting that h is cycle-free. Similarly $m_2 \not\rightsquigarrow m_1$. $\square$

6.2. LEMMA. *Let g_1, g_2 be graphs. Suppose $g_1 \succeq g_2$.*

(i) *Let $n, m \in g_2$ with $n \neq m$. Say $\varphi^{-1}(n) = \{n_1, \ldots, n_k\}$. Then for any $m_1, \ldots, m_k \in \varphi^{-1}(m)$*

$$g_1[n_1 := m_1, \ldots, n_k := m_k] \succeq_\varphi g_2[n := m].$$

(ii) *Let $E = \langle h, H, \mu \rangle$ be an extension of g_2, and let $E_1 = \langle h, H, \mu_1 \rangle, \ldots, E_k = \langle h, H, \mu_k \rangle$ be extensions of g_1 such that $\mu = \varphi \circ \mu_i$. Then*

$$g_1 + E_1 + \cdots + E_k \succeq g_2 + E.$$

PROOF. (i) First note that the left-hand side is well-defined since $\varphi^{-1}(n)$ and $\varphi^{-1}(m)$ are disjoint. Set $g_1' = g_1[n_1 := m_1, \ldots, n_k := m_k]$, $g_2' = g_2[n := m]$. Let $p \in g_1'$. Then one has $\varphi(args_{g_1'}(p)_i) = args_{g_2'}(\varphi(p))_i$, as is shown by the following case distinction.

Case 1. $args_{g_1}(p)_i = n_j$ for some j. Then $args_{g_1'}(p)_i = m_j$. Moreover $args_{g_2}(\varphi(p))_i = \varphi(args_{g_1}(p)_i) = n$, so

$$\begin{aligned} \varphi(args_{g_1'}(p)_i) &= m \\ &= [n \mapsto m](args_{g_2}(\varphi(p))_i) \\ &= args_{g_2'}(\varphi(p))_i. \end{aligned}$$

Case 2. $args_{g_1}(p)_i \neq n_j$ for all j. Then $\varphi(args_{g_1}(p)_i) \neq n$. Hence

$$\begin{aligned} \varphi(args_{g_1'}(p)_i) &= \varphi(args_{g_1}(p)_i) \\ &= args_{g_2}(\varphi(p))_i \\ &= args_{g_2'}(\varphi(p))_i. \end{aligned}$$

(ii) Set $g_1' = g_1 + E_1 + \cdots + E_k$, $g_2' = g_2 + E$. Note that $N_1' = N_1 \cup^+ H \cup^+ \cdots \cup^+ H$, and $N_2' = N_2 \cup^+ H$. Let $\varphi^\sharp$ be the canonical extension of φ from N_1' to N_2'. Then $\varphi^\sharp$ is a homomorphism from g_1' to g_2'. □

6.3. Proposition. *Let $g_1 \succeq_\varphi g_2$, and $n \in g_2$. Say $\varphi^{-1}(n) = \{n_1, \ldots, n_k\}$. Let $E = \langle h, H, \mu\rangle$ be an extension of g_2, and $E_1 = \langle h, H, \mu_1\rangle, \ldots, E_k = \langle h, H, \mu_k\rangle$ be extensions of g_1, such that $\mu = \psi \circ \mu_i$ for all i. Suppose $\mathrm{r}(E_i) \neq n_j$ for all $i \neq j$. Then*

$$g_1[n_1 := E_1] \cdots [n_k := E_k] \succeq g_2[n := E].$$

Proof.

$$\begin{aligned} & g_1[n_1 := E_1] \cdots [n_k := E_k] = \\ & \quad = \mathrm{GC}((g_1 + E_1 + \cdots + E_k)[n_1 := \mathrm{r}(E_1), \ldots n_k := \mathrm{r}(E_k)]), \\ & \qquad \text{by propositions 4.30 and 4.29 (iii)} \\ & \quad \succeq \mathrm{GC}((g + E)[n := m]), \quad \text{by lemma 6.2. } \square \end{aligned}$$

6.4. Lemma. *Let $g' \succeq_\varphi g$ and $s \subseteq g$. Define*

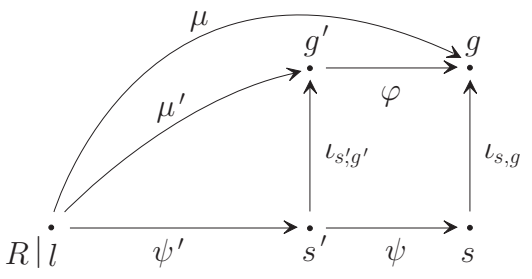
$$s' = g' \rtimes \varphi^{-1}(\iota(s^\circ)).$$

Then $s' \succeq s$.

Proof. Straightforward. □

6.5. Lemma. *Let $g' \succeq_\varphi g$, and let $\Delta = \langle R, \mu\rangle$ be a redex in g. Let $m \in \varphi^{-1}(\mu(l))$. Then there exists a μ' such that $\Delta' = \langle R, \mu'\rangle$ is a redex in g', $\mu'(l) = m$, and $\mu = \varphi \circ \mu'$.*

Proof. Using lemma 4.9, determine $s \subseteq g \mid \mu(l)$ such that $(R \mid l) \succeq s$. By lemma 6.4 there exist $s' \subseteq g' \mid m$ and ψ with $s \preceq_\psi s'$. Since $R \mid l$ is a tree one also has $(R \mid l) \succeq_{\psi'} s'$ for some ψ'. Define $\mu' = \iota_{s',g'} \circ \psi'$. Then indeed $\mu = \varphi \circ \mu'$, by uniqueness of homomorphisms.



□

6.6. Proposition. *Let $g_1 \succeq g_2$. Suppose $g_2 \xrightarrow[\mathcal{R}]{} h_2$. Then $g_1 \xrightarrow[\mathcal{R}]{} h_1$ for some $h_1 \succeq h_2$.*

Proof. Say $g_1 \succeq_\varphi g_2$ and $g_2 \xrightarrow[\mathcal{R}]{\Delta} h_2$ with $\Delta = \langle R, \mu\rangle$. Set $n = \mu(l)$. Say $\varphi^{-1}(n) = \{n_1, \ldots, n_k\}$. Using lemma 6.5, determine $\Delta_1 = \langle R, \mu_1\rangle, \ldots, \Delta_k = \langle R, \mu_k\rangle$ such that $\mu_1(l) = n_1, \ldots, \mu_k(l) = n_k$. Then $\mathrm{r}(\Delta_i) \neq \mu_j(l)$ for all $i \neq j$, by lemma 6.1. Take

$$h_1 = g_1[\mu_1(l) := \Delta_1] \cdots [\mu_k(l) := \Delta_k].$$

Then $h_1 \succeq h_2$ by proposition 6.3. □

7. Copying

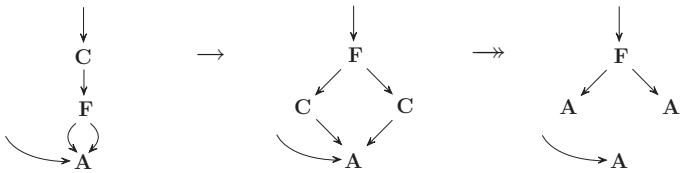
Extending graph rewriting with a copying mechanism can be done in several ways. Below some of these are discussed.

Intuitively, copying a graph causes a new graph to be created that has the same structure as, but no nodes in common with the original graph. Let **C** be a unary symbol in Σ. A copy node in a graph g is a node with symbol **C**. The intended meaning is that evaluation of a copy node leads to duplication of the graph it refers to. We want to describe how such a **C**-node can be handled.

The first possibility is to extend an existing TGRS $\mathcal{T} = \langle \mathcal{G}, \mathcal{R} \rangle$ to a system $\mathcal{T}^{\mathrm{C}}$ by adding new rewrite rules for copy reduction to $\mathcal{R}$, thus obtaining $\mathcal{R}^{\mathrm{C}}$. This can be done in various ways, the simplest of which is adding

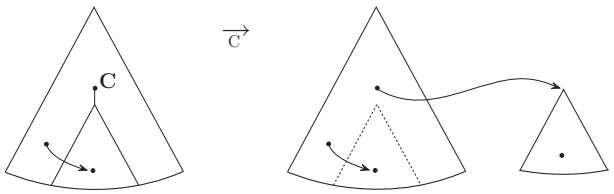
$$\mathbf{C}(\mathbf{F}(x_1, \ldots, x_n)) \to \mathbf{F}(\mathbf{C}(x_1), \ldots, \mathbf{C}(x_n))$$

for each symbol **F** with arity n in $\mathcal{G}$. Although this extension can be described easily, there are several disadvantages. First of all, the number of extra rewrite rules may be infinite if in $\mathcal{G}$ infinitely many symbols are used. Furthermore, mixing copy rules with the reduction rules in $\mathcal{R}$ may destroy properties of $\mathcal{R}$-reduction such as confluence (see section 9), or at least make it very difficult to check whether known properties of $\mathcal{T}$ extend to $\mathcal{T}^{\mathrm{C}}$. This is reinforced by the fact that copying is done stepwise and therefore cannot be considered as a basic action such as redirection or extension. Another unwanted effect is that sharing cannot be maintained, as is illustrated below. The loose arrows indicate edges coming in from the surrounding graph.



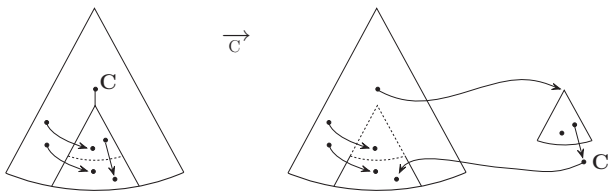
Thus shared redexes below **F** are duplicated. This last drawback could in principle be overcome by using more sophisticated rewrite rules.

The second possibility is the addition of a new copy mechanism to graph rewriting. A rude solution involves replacing the **C**-node by a copy of the whole subgraph it refers to. This method is called *eager copying*. It is visualized in the following figure.



Now all the sharing below the copy node remains intact. However, all redexes below that node are duplicated. This involves the 'duplication of work'. It may therefore be convenient to duplicate only some redexes of the subgraph in question and postpone the rest of the copy action. This is called *lazy copying*. New copy nodes are added

where the copy procedure has been interrupted. In a picture this looks as follows.



Of course, this mechanism is more complicated to describe than the previous one. Moreover, if one uses a reduction strategy to select a specific reduction order matters become complicated because one has to decide at which point to interrupt the copying. We will go into this in section 10.

The rest of this section is devoted to a formal description of lazy copying.

7.1. Definition. Let g be a graph.

(i) Let $n \in g$ and $s \subseteq g \mid n$. Note that $\langle s, \iota_{s,g} \rangle$ is an extension of g. The *s-expansion of g at n* (notation $g[n := s]$) is defined by

$$g[n := s] = g[n := \langle s, \iota_{s,g} \rangle].$$

(ii) g' is an *expansion* of g if $g' = g[n := s]$ for some n and s.

7.2. Lemma. *Let $s \subseteq g \mid n$. Then*

$$g \preceq g[n := s].$$

Proof. Set $g^+ = g[n := s]$. Write $N_{g^+} = N \cup S^*$, where S^* are the new nodes originating from s. Define

$$\begin{array}{rcll} \varphi(n) & = & n & \text{if } n \in N, \\ \varphi(n^*) & = & n & \text{for } n^* \in S^*. \end{array}$$

Then $g \preceq_\varphi g^+$. $\square$

7.3. Definition. Let $\mathbf{C}$ be a special symbol in Σ with arity 1.

(i) A node $n \in g$ is a *copy node* if $symb_g(n) = \mathbf{C}$. The set of copy nodes of g is denoted by $\mathrm{C}(g)$.

(ii) The definition of *term graph rewrite rule* (4.3) is extended with the following requirement.

(3) $\mathrm{C}(g \mid l) = \emptyset$, if $R = \langle g, l, r \rangle$ is a rule.

This requirement states that only the right-hand side of a rewrite rule may contain copy nodes. Thus copy nodes may not be used as part of a pattern.

7.4. Definition. Let g be a graph, and $M \subseteq N_g$. The *copy indirection* of M in g (notation $\mathrm{CI}(g, M)$) is the graph g' that is obtained from g by redirecting each node

$n \in M$ to a new copy node with n as argument. More formally

$$\begin{array}{rcll}
N_{g'} & = & N_g \cup^+ M; & \\
symb_{g'}(n) & = & symb_g(n) & \text{if } n \in g, \\
 & = & \mathbf{C} & \text{otherwise;} \\
args_{g'}(n)_i & = & args_g(n)_i & \text{if } n \in g \text{ and } args_g(n)_i \notin M, \\
 & = & (args_h(n)_i)^* & \text{if } n \in g \text{ and } args_g(n)_i \in M, \\
args_{g'}(n^*)_i & = & n & \text{for } n^* \in M^*; \\
r_{g'} & = & r_g & \text{if } r_g \notin M, \\
 & = & {r_g}^* & \text{if } r_g \in M.
\end{array}$$

NOTATION. If n is a unary node in g then $args_g(n)_1$ is written as $succ(n)$.

7.5. DEFINITION. (i) A *copy redex* in a graph g is a pair $\langle n, s\rangle$ where
(1) $n \in \mathrm{C}(g)$
(2) $s \subseteq g \mid succ(n)$ and $s^\bullet \neq \emptyset$.
(ii) Let $\Delta = \langle n, s\rangle$ be a copy redex of g. The result g' of *contracting* Δ in g is defined by the replacement

$$g' = g[n := \mathrm{CI}(s, s^\circ)],$$

which is shorthand for $g[n := \langle \mathrm{CI}(s, s^\circ), \iota\rangle]$.
(iii) We write $g \xrightarrow[\mathrm{C}]{\Delta} g'$ to indicate that g *copy-reduces* to g' via redex Δ.

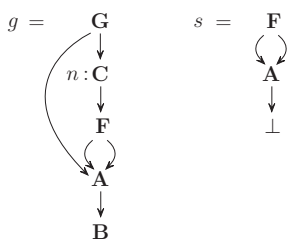
7.6. DEFINITION. A *copy term graph rewrite system* (C-TGRS) is a structure

$$\mathcal{T} = \langle \mathcal{G}, \mathcal{R}\rangle$$

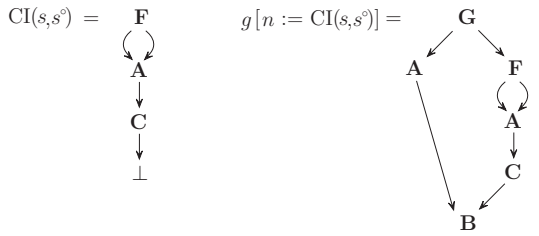
where $\mathcal{G} \subseteq \mathbb{G}$ is a set of graphs, closed under $\mathcal{R}$-reduction and C-reduction.

7.7. REMARK. Notions introduced for TGRS's (such as left-linearity and orthogonality) are used for C-TGRS's as well by considering them as ordinary TGRS's for a moment.

7.8. EXAMPLE. Let g, n and $s \subseteq g \mid succ(n)$ be as follows.



The next figure displays the copy indirection of s and the result of contracting copy redex $\langle n, s\rangle$ in g.



8. Analyzing copy reduction

Because lazy copying is a complicated concept, we will decompose copy reduction into three operations, namely

(1) partial unraveling (see section 3),
(2) addition of **C**-nodes,
(3) elimination of **C**-nodes.

The main part of this section addresses the **C**-elimination mechanism and its interaction with the three basic graph operations mentioned in section 4, and with unraveling. Then it is shown that copy reduction can indeed be splitted in the above-mentioned way.

The elimination of **C**-nodes can be viewed as a redirection. This is made precise in the following.

8.1. Definition. Let g be a rooted graph, and D be a collection of of nodes of g. A path p in g is *within* D (notation $p \subseteq D$) if $n \in D$ for all $n \in p$. D is *cycle-free* (*in* g) if all the paths of g within D are a-cyclic.

8.2. Remark. Let g be graph. Let $D \subseteq g$ be a finite set of unary nodes. Then every a-cyclic path p in D is uniquely determined by its begin and end points, for each node of p has exactly one successor. Consequently, if D is cycle-free then for any $n \in g$ there exists exactly one $p : n \rightsquigarrow m$ such that

$$m \notin D,$$
$$\forall m' \in p \quad [m' \neq m \;\Rightarrow\; m' \in D].$$

The latter expression is abbreviated to $p - m \subseteq D$.

In the sequel we will only consider finite sets D consisting of nodes with arity 1. Remark 8.2 enables us to define the following.

8.3. Definition. Let g be a graph, and $D \subseteq \mathrm{C}(g)$ cycle-free. Define the redirection $\mathrm{e}_{D,g}$ as follows. Let $n \in g$. Then $\mathrm{e}_{D,g}(n)$ is the uniquely determined $m \notin D$ such that $p : n \rightsquigarrow m$ for some p with $p - m \subseteq D$. Then $\mathrm{e}_{D,g}$ is obviously projective, and $\mathrm{Fld}(\mathrm{e}_{D,g}) = D$.

Now the actual elimination of **C**-nodes in g is obtained by applying the corresponding redirection operation to g.

8.4. DEFINITION. Let g be a graph and $D \subseteq \mathrm{C}(g)$. $\mathrm{E}_D(g)$ denotes the graph that is obtained by deleting all copy nodes of D from g, i.e. $\mathrm{E}_D(g) = \mathrm{e}_{D,g}(g)$.

8.5. LEMMA. *Let g be a graph, $D_1, D_2 \subseteq \mathrm{C}(g)$ such that $D_1 \cup D_2$ is cycle-free. Then*

$$\mathrm{e}_{D_1 \cup D_2, g} = \mathrm{e}_{D_2, \mathrm{E}_{D_1}(g)} \circ \mathrm{e}_{D_1, g}.$$

PROOF. Suppose $\mathrm{e}_{D_1 \cup D_2, g}(n) = m$, say $p : n \rightsquigarrow m$ with $p - m \subseteq D_1 \cup D_2$. Set $h = \mathrm{E}_{D_1}(g)$. Then for each $n' \in p - m$ one has either $\mathrm{e}_{D_1,g}(n') \in D_2$ or $\mathrm{e}_{D_1,g}(n') = n$. Moreover $\mathrm{e}_{D_1}(g)(m) = m$. So in h there exists a path $p' : \mathrm{e}_{D_1,g}(n) \rightsquigarrow m$ such that $p' - m \subseteq D_2$. Hence $\mathrm{e}_{D_2,h} \circ \mathrm{e}_{D_1,g}(n) = m$. □

8.6. PROPOSITION. *Let g be a graph, $D_1, D_2 \subseteq \mathrm{C}(g)$ such that $D_1 \cup D_2$ is cycle-free. Then*

$$\mathrm{E}_{D_1 \cup D_2}(g) = \mathrm{E}_{D_2}(\mathrm{E}_{D_1}(g)).$$

PROOF. By lemma 8.5. □

8.7. LEMMA. *Let g be a graph, $D \subseteq \mathrm{C}(g)$ such that D is finite and cycle-free. Then*

$$\mathrm{e}_{D,g} = [d_1 \mapsto \mathrm{e}_{D,g}(d_1), \ldots, d_k \mapsto \mathrm{e}_{D,g}(d_k)]$$

if $D = \{d_1, \ldots, d_k\}$.

PROOF. By lemma 4.14. □

The interaction between $\mathrm{e}_{D,g}$ and extension and garbage collection respectively is straightforward.

8.8. LEMMA. *Let g be a graph, and let $E = \langle h, H, \mu \rangle$ be an extension of g. Let D be a collection of copy nodes that is cycle-free in $g +_\mu \langle h, H \rangle$. Then*

(i) $D \cap h = \emptyset \;\Rightarrow\; \mathrm{e}_{D, g +_\mu \langle h,H \rangle} = \mathrm{e}_{D,g}$.

(ii) $D \cap g = \emptyset \;\Rightarrow\; \mathrm{e}_{D, g +_\mu \langle h,H \rangle} = \mathrm{e}_{D,h}$.

PROOF. (i) Since D does not contain any node from h it follows that each path in $g +_\mu \langle h, H \rangle$ within D is also a path in g.

(ii) Similarly. □

8.9. LEMMA. *Let g be a graph, and $D \subseteq \mathrm{C}(g)$ cycle-free.*

(i) *If $\mathrm{e}_{D,g}$ is applicable to $g \mid n$ then $\mathrm{e}_{D,g|n} = \mathrm{e}_{D,g}$.*

(ii) $\mathrm{GC}(\mathrm{e}_{D,\mathrm{GC}(g)}(\mathrm{GC}(g)) = \mathrm{GC}(\mathrm{e}_{D,g}(g))$.

PROOF. (i) Every path in $g \mid n$ is also a path in g.

(ii) By proposition 4.29 (i) one has $\mathrm{GC}(\mathrm{e}_{D,\mathrm{GC}(g)}(\mathrm{GC}(g)) = \mathrm{GC}(\mathrm{e}_{D,\mathrm{GC}(g)}(g))$. From (i) and lemma 4.28 (i, ii) it follows that

$$\mathrm{e}_{D,g|n}(g \mid n) \mid \mathrm{e}_{D,g|n}(n) = \mathrm{e}_{D,g}(g) \mid \mathrm{e}_{D,g}(n)$$

for all nodes $n \in g$. Hence $\mathrm{GC}(\mathrm{e}_{D,\mathrm{GC}(g)}(\mathrm{GC}(g)) = \mathrm{GC}(\mathrm{e}_{D,g}(g))$ by taking $n = r_g$. □

8.10. Remark. Let g be a graph and D a cycle-free collection of nodes. Then, after garbage collection, $\mathrm{e}_{D,g}(g)$ and D do not have any nodes in common, i.e. $D \cap \mathrm{GC}(\mathrm{e}_{D,g}(g)) = \emptyset$.

Now the relation between **C**-elimination and $\mathcal{R}$-reduction is investigated.

8.11. Lemma. *Let g be a graph, $D \subseteq g$ be cycle-free in g, and $n \in g$ such that $n \notin D$. Then for any $m \neq n$ such that D is cycle free in $g[n := m]$*

(i) $\mathrm{e}_{D,g}(m) \neq n$;

(ii) $\mathrm{e}_{D,g}(a) \neq n \;\Rightarrow\; \mathrm{e}_{D,g[n:=m]}(a) = \mathrm{e}_{D,g}(a)$;

(iii) $\mathrm{e}_{D,g}(a) = n \;\Rightarrow\; \mathrm{e}_{D,g}(m) = \mathrm{e}_{D,g[n:=m]} \circ [n \mapsto m](a)$.

Proof. (i) Suppose $\mathrm{e}_{D,g}(m) = n$, say $p : m \rightsquigarrow n$ with $p - n \subseteq D$. Note that $|p| \geq 1$ since $m \neq n$. But then in $g[n := m]$ one has $m \rightsquigarrow m$ via some path $p' \subseteq D$ with $|p'| \geq 1$, contradicting the assumption that D is cycle-free in $g[n := m]$.

(ii) Suppose $\mathrm{e}_{D,g}(a) \neq n$. Set $k = \mathrm{e}_{D,g}(a)$, say $p : a \rightsquigarrow k$ in g with $p - k \subseteq D$. Then $n \notin p$ since $n \notin D$ and $n \neq k$. So p is a path in $g[n := m]$ (see remark 4.17). Hence $\mathrm{e}_{D,g[n:=m]}(a) = k$.

(iii) The case $a = n$ is handled by (i) and (ii). Now assume $a \neq n$. Say $p : a \rightsquigarrow n$ in g. There also exists a path $p : m \rightsquigarrow \mathrm{e}_{D,g}(m)$ in g. Then $p * p'$ is a path in $g[n := m]$ (see remark 4.17). Note that $(p * p') - \mathrm{e}_{D,g}(m) \subseteq D$. Hence $\mathrm{e}_{D,g[n:=m]}(a) = \mathrm{e}_{D,g}(m)$. □

8.12. Proposition. *Let g be a graph, $D \subseteq g$ be cycle-free in g, and $n \in g$ such that $n \notin D$. Then for any m such that D is cycle free in $g[n := m]$ one has*

$$[n \mapsto \mathrm{e}_{D,g}(m)] \circ \mathrm{e}_{D,g} = \mathrm{e}_{D,g[n:=m]} \circ [n \mapsto m].$$

Proof. If $m = n$ the equality trivially holds. Moreover lemma 8.11 is used in case $m \neq n$. □

8.13. Corollary. *Let g be a graph, $m, n \in g$, and let $D \subseteq g$ be cycle-free in both g and $g[n := m]$. If $n \notin D$, then*

$$\mathrm{E}_D(g[n := m]) = \mathrm{E}_D(g)[n := \mathrm{e}_{D,g}(m)].$$

Proof. By proposition 8.12. □

8.14. Proposition. *Let $g \underset{\mathcal{R}}{\rightarrow} h$. Let $D \subseteq \mathrm{C}(h)$ be cycle-free. Then*

$$\mathrm{GC}(\mathrm{E}_D(g)) \underset{\mathcal{R}}{\rightarrow} \mathrm{GC}(\mathrm{E}_D(h)).$$

Proof. It will be shown that

$$\mathrm{GC}(\mathrm{e}_{D,g}(g)) \underset{\mathcal{R}}{\rightarrow} \mathrm{GC}(\mathrm{e}_{D,h}(h)).$$

Say $g \overset{\Delta}{\underset{\mathcal{R}}{\rightarrow}} h$ with $\Delta = \langle R, \mu \rangle$. Then

$$h = \mathrm{GC}((g + \Delta)[\mu(l) := \mathrm{r}(\Delta)]).$$

Observe that $\mu(R \mid l)^\bullet \cap D = \emptyset$ since $(R \mid l)^\bullet$ is **C**-free. So $\mathrm{e}_{D,g}(n) = n$ for any $n \in \mu(R \mid l)^\bullet$. Then by lemma 4.27 one has $\mathrm{e}_{D,g} \circ \mu : (R \mid l)^\bullet \xrightarrow[\mathrm{m}]{} \mathrm{e}_{D,g}(g)$. Write $\mathrm{e}_{D,g}(\Delta) = \langle R, \mathrm{e}_{D,g} \circ \mu \rangle$. Then $\mathrm{e}_{D,g}(\Delta)$ is a redex in $\mathrm{e}_{D,g}(g)$. Note that $\mathrm{r}(\mathrm{e}_{D,g}(\Delta)) = \mathrm{e}_{D,g}(\mathrm{r}(\Delta))$, and $\mathrm{e}_{D,g}(\mu(l)) = \mu(l)$. Contracting $\mathrm{e}_{D,g}(\Delta)$ in $\mathrm{GC}(\mathrm{e}_{D,G}(g))$ results in

$$
\begin{aligned}
&\mathrm{GC}(\ (\ \mathrm{GC}(\mathrm{e}_{D,g}(g)) + \mathrm{e}_{D,g}(\Delta)\ )[\mu(l) := \mathrm{r}(\mathrm{e}_{D,g}(\Delta))]\ ) \\
&\quad = \quad \mathrm{GC}(\ (\ \mathrm{e}_{D,g}(g) + \mathrm{e}_{D,g}(\Delta)\ )[\mu(l) := \mathrm{r}(\mathrm{e}_{D,g}(\Delta))]\ ), \\
&\qquad\qquad \text{by proposition 4.29 (iii)} \\
&\quad = \quad \mathrm{GC}(\ (\ \mathrm{e}_{D,g}(g + \Delta)\ )[\mu(l) := \mathrm{r}(\mathrm{e}_{D,g}(\Delta))]\ ), \\
&\qquad\qquad \text{by proposition 4.30} \\
&\quad = \quad \mathrm{GC}(\ (\ \mathrm{e}_{D,g}(g + \Delta)\ )[\mu(l) := \mathrm{e}_{D,g}(\mathrm{r}(\Delta))]\ ) \\
&\quad = \quad \mathrm{GC}(\ (\ \mathrm{e}_{D,\mathrm{GC}(g+\Delta)}(g + \Delta)\ )[\mu(l) := \mathrm{e}_{D,g}(\mathrm{r}(\Delta))]\ ), \\
&\qquad\qquad \text{by lemma 4.26 (ii)} \\
&\quad = \quad \mathrm{GC}(\ \mathrm{e}_{D,\mathrm{GC}(g+\Delta)[\mu(l):=\mathrm{r}(\Delta)]}(\ (g + \Delta)[\mu(l) := \mathrm{r}(\Delta)]\ )\ ), \\
&\qquad\qquad \text{by proposition 8.12} \\
&\quad = \quad \mathrm{GC}(\ \mathrm{e}_{D,\mathrm{GC}(g+\Delta)[\mu(l):=\mathrm{r}(\Delta)]}(\ \mathrm{GC}(\ (g + \Delta)[\mu(l) := \mathrm{r}(\Delta)]\ )\ )\ ), \\
&\qquad\qquad \text{by proposition 4.29 (i)}, \\
&\quad = \quad \mathrm{GC}(\ \mathrm{e}_{D,h}(h)\ ).\ \square
\end{aligned}
$$

The relation between partially unraveling a graph and the elimination of copy nodes is straightforward.

8.15. Lemma. *Let $g \succeq_\varphi h$. Then for any finite cycle-free $D \subseteq \mathrm{C}(h)$*

$$\mathrm{e}_{\varphi^{-1}(D),g}(g) \succeq_\varphi \mathrm{e}_{D,h}(h).$$

Proof. Write $g' = \mathrm{e}_{\varphi^{-1}(D),g}(g)$ and $h' = \mathrm{e}_{D,h}(h)$. Let $n \in g$. Then $args_{g'}(n)_i = \mathrm{e}_{\varphi^{-1}(D),g}(args_g(n)_i)$. Determine $p : args_g(n)_i \rightsquigarrow args_{g'}(n)_i$ in g, such that $p - args_{g'}(n)_i \subseteq \varphi^{-1}(D)$ and $args_{g'}(n)_i \notin \varphi^{-1}(D)$. Then $\varphi(p) : \varphi(args_g(n)_i) \rightsquigarrow \varphi(args_{g'}(n)_i)$ in h, with $\varphi(p) - \varphi(args_{g'}(n)_i) \subseteq D$, and $\varphi(args_{g'}(n)_i) \notin D$. Therefore

$$
\begin{aligned}
\varphi(args_{g'}(n)_i) &= \mathrm{e}_{D,h}(\varphi(args_g(n)_i)) \\
&= \mathrm{e}_{D,h}(args_h(\varphi(n))_i), \quad \text{since } \varphi \text{ is a homomorphism} \\
&= args_{h'}(\varphi(n))_i,
\end{aligned}
$$

so $g' \succeq_\varphi h'$. $\square$

8.16. Proposition. *Let $g \succeq_\varphi h$. Let $D \subseteq \mathrm{C}(h)$ be cycle-free. Then*

$$\mathrm{GC}(\mathrm{E}_{\varphi^{-1}(D)}(g)) \succeq_\varphi \mathrm{GC}(\mathrm{E}_D(h)).$$

Proof. By lemma 8.15. $\square$

Some technical work has to be done in order to split copy reduction into more basic actions.

8.17. Lemma. *Let g be a graph, and $\langle h, \mu\rangle$ an extension of g. Furthermore, let $D \subseteq h^\bullet$ such that $succ(D) \subseteq h^\bullet$. Then*

$$\mathrm{e}_{D,g+_\mu h}(g +_\mu h) = g +_\mu \mathrm{e}_{D,h}(h).$$

Proof. From $D \subseteq h^\bullet$ and $succ(D) \subseteq h^\bullet$ if follows that the restrictions of $\mu \circ \mathrm{e}_{D,h}$ to h° and $g +_\mu h$ are μ and $\mathrm{e}_{D,h}$ respectively. Then

$$\begin{aligned} g +_\mu \mathrm{e}_{D,h}(h) &= \mu \circ \mathrm{e}_{D,h}(g +_{\mu\circ\mathrm{e}_{D,h}} h), \quad \text{by proposition 4.30} \\ &= \mathrm{e}_{D,h}(g +_\mu h) \\ &= \mathrm{e}_{D,g+_\mu h}(g +_\mu h), \quad \text{by lemma 8.8.} \ \square \end{aligned}$$

8.18. Proposition. *Let g be a graph, $\langle h, \mu\rangle$ an extension of g. Furthermore, let $D \subseteq h^\bullet$ such that $succ(D) \subseteq h^\bullet$. Then $\mathrm{E}_D(g +_\mu h) = g +_\mu \mathrm{E}_D(h)$.*

Proof. By lemma 8.17. $\square$

8.19. Lemma. *Let g be a graph, $D \subseteq g$. Then $\mathrm{GC}(\mathrm{E}_{D^*}(\mathrm{CI}(g, D))) = \mathrm{GC}(g)$.*

Observe that g contains the nodes of D as well as of D^*. Hence it is necessary to indicate nodes of D^* explicitly.

Proof. According to remark 8.10 we do not have to consider nodes of D^*. Set $h = \mathrm{GC}(\mathrm{E}_{D^*}(\mathrm{CI}(g, D)))$. Notice that $\mathrm{e}_{D^*,h}(n) = succ(n)$ if $n \in D$.

Case 1. $args_g(n)_i \in D$ or $args_g(n)_i \in D^*$. Then

$$args_h(n)_i = succ(args_g(n)_i{}^*) = args_g(n)_i.$$

Case 2. Otherwise. Then

$$args_h(n)_i = succ(args_g(n)_i) = args_g(n)_i. \ \square$$

8.20. Lemma. *Let g be a graph with a unary root. Then*

$$\mathrm{GC}(\mathrm{E}_{\{r_g\}}(g)) = g \mid succ(r_g).$$

Proof. By lemma 8.7, $\mathrm{E}_{\{r_g\{}(g)$ can be written as $g[r_g := succ(r_g)]$. Now the equality of both graphs is obvious. $\square$

8.21. Lemma. *Let g be a graph, n a unary node of g. Furthermore, let $s \subseteq g \mid n$ and $s_n = s \mid succ(n)$. Then*

$$\mathrm{GC}(\mathrm{E}_{\{r_s\}}(g[n := s])) = \mathrm{GC}(g[n := s_n]).$$

Proof. Note that $\iota_{s,g} = \iota_{s_n,g}$. Set $\iota = \iota_{s,g}$. Then

$$\begin{aligned} &\mathrm{GC}(\mathrm{E}_{\{r_s\}}(g[n := \langle s, \iota\rangle])) \\ &= \mathrm{GC}(\mathrm{E}_{\{r_s\}}((g +_\iota s)[n := r_s])) \\ &= \mathrm{GC}(\mathrm{E}_{\{r_s\}}(g +_\iota s)[n := \mathrm{e}_{\{r_s\},g+_\iota s}(r_s)], \quad \text{by corollary 8.13} \\ &= \mathrm{GC}(g +_\iota \mathrm{GC}(\mathrm{E}_{\{r_s\}}(s)))[n := succ(r_s)], \\ &\qquad \text{by propositions 8.18 and 4.29 (iv)} \\ &= \mathrm{GC}(g +_\iota s \mid succ(r_s))[n := succ(r_s)], \quad \text{by lemma 8.20} \\ &= \mathrm{GC}(g[n := \langle s_\mathrm{n}, \iota\rangle]). \ \square \end{aligned}$$

8.22. Lemma. *Let g be a graph, $n \in g$. Furthermore let $s \subseteq g \mid n$ such that $s^\bullet \neq \emptyset$. Then*

$$\mathrm{GC}(\mathrm{E}_{(s^\circ)^*}(g[n := \mathrm{CI}(s, s^\circ)])) = \mathrm{GC}(g[n := s]).$$

Proof. Write $h = g +_\iota \mathrm{CI}(s, s^\circ)$. From $s^\bullet \neq \emptyset$ it follows that the redirections $\mathrm{e}_{(s^\circ)^*,h}$ and $[n \mapsto r_s]$ are independent. Furthermore, notice that $\mathrm{e}_{(s^\circ)^*,h}(n^*) = \iota(n)$. Then $\mathrm{e}_{(s^\circ)^*,h}$ can be written as $[{s_1}^* \mapsto \iota(s_1), \ldots, {s_k}^* \mapsto \iota(s_k)]$ where $(s^\circ)^* = \{s_1, \ldots, s_k\}$. Since ι is injective it has an inverse ι^{-1} on $\iota(s^\circ)$. Hence $\iota^{-1} \circ \mathrm{e}_{(s^\circ)^*,h} = [{s_1}^* \mapsto s_1, \ldots, {s_k}^* \mapsto s_k]$ which is the same as $\mathrm{e}_{(s^\circ)^*,\mathrm{CI}(s,s^\circ)}$. Then

$$\begin{aligned}
&\mathrm{GC}(\mathrm{E}_{(s^\circ)^*}(g[n := \langle \mathrm{CI}(s, s^\circ), \iota\rangle])) \\
&= \mathrm{GC}(\mathrm{E}_{(s^\circ)^*}(h)[n := r_s])) \\
&= \mathrm{GC}((\mathrm{e}_{(s^\circ)^*,h}(h))[n := r_s]), \\
&\qquad \text{since } [n \mapsto r_s] \text{ and } \mathrm{e}_{(s^\circ)^*,h} \text{ are independent} \\
&= \mathrm{GC}((g +_\iota \mathrm{GC}(\iota^{-1} \circ \mathrm{e}_{(s^\circ)^*,h}(\mathrm{CI}(s, s^\circ)))[n := r_s]), \\
&\qquad \text{by propositions 4.30 and 4.29 (iv)} \\
&= \mathrm{GC}((g +_\iota \mathrm{GC}(\mathrm{e}_{(s^\circ)^*,\mathrm{CI}(s,s^\circ)}(\mathrm{CI}(s, s^\circ)))[n := r_s]), \\
&\qquad \text{by the remark made above} \\
&= \mathrm{GC}((g +_\iota s)[n := r_s]), \quad \text{by lemma 8.19} \\
&= \mathrm{GC}(g[n := \langle s, \iota\rangle]). \ \square
\end{aligned}$$

For convenience some new reduction relations are introduced.

8.23. Definition. The reduction relations $\underset{\preceq}{\to}$, $\underset{-}{\to}$ and $\underset{+}{\to}$ are defined by

$$\begin{aligned}
g \underset{\preceq}{\to} h &\Leftrightarrow g \preceq h, \\
g \underset{-}{\to} h &\Leftrightarrow h = \mathrm{GC}(\mathrm{E}_D(g)) \quad \text{for some nonempty } D \subseteq \mathrm{C}(g), \\
g \underset{+}{\to} h &\Leftrightarrow \mathrm{C}(g) \neq \emptyset \text{ and } g = \mathrm{GC}(\mathrm{E}_D(h)) \quad \text{for some nonempty } D \subseteq \mathrm{C}(h).
\end{aligned}$$

8.24. Proposition. $g \underset{\mathrm{C}}{\to} g' \ \Rightarrow \ g \underset{\preceq}{\to} \cdot \underset{+}{\to} \cdot \underset{-}{\to} g'$.

Proof. Let $\langle n, s\rangle$ be the copy redex of g that is reduced in order to obtain g'. Hence $s \subseteq g \mid succ(n)$ and $g' = \mathrm{GC}(g +_\iota \mathrm{CI}(s, s^\circ)[n := r_s])$. Let $s_\mathrm{c} \subseteq g \mid n$ such that $s = s_\mathrm{c} \mid succ(n)$. Then

$$\begin{aligned}
g &\underset{\preceq}{\to} \mathrm{GC}(g[n := s_\mathrm{c}]), \quad \text{by lemma 7.2} \\
&\underset{+}{\to} \mathrm{GC}(g[n := \mathrm{CI}(s_\mathrm{c}, s_\mathrm{c}^\circ)]), \quad \text{by lemma 8.22} \\
&\underset{-}{\to} \mathrm{GC}(g[n := \mathrm{CI}(s, s^\circ)]), \quad \text{by lemma 8.21} \\
&= g'. \ \square
\end{aligned}$$

9. Confluence

This section starts with the introduction of some standard terminology.

9.1. Definition. (i) A *reduction relation* on $\mathcal{G}$ is just a binary relation $\boldsymbol{R}$ on $\mathcal{G}$.

(ii) If $\boldsymbol{R}_1, \boldsymbol{R}_2$ are reduction relations, then $\boldsymbol{R}_1\boldsymbol{R}_2$ is $\boldsymbol{R}_1 \cup \boldsymbol{R}_2$.
(iii) A graph g *$\boldsymbol{R}$-reduces in one step* to g' (notation $g \underset{\boldsymbol{R}}{\rightarrow} g'$) if $(g, g') \in \boldsymbol{R}$.

9.2. DEFINITION. Let $\boldsymbol{R}$ be a reduction relation. Then $\boldsymbol{R}$ induces the following binary relations.
(i) The relation $\underset{\boldsymbol{R}}{\twoheadrightarrow}$ (*$\boldsymbol{R}$-reduction*) is the reflexive transitive closure of $\underset{\boldsymbol{R}}{\rightarrow}$.
(ii) $\underset{\boldsymbol{R}}{\rightarrow}^+$ (*progressive $\boldsymbol{R}$-reduction*) is the transitive closure of $\underset{\boldsymbol{R}}{\rightarrow}$.
(iii) $\underset{\boldsymbol{R}}{\rightarrow}^{\equiv}$ (*$\boldsymbol{R}$-reduction in zero or one step*) is the reflexive closure of $\underset{\boldsymbol{R}}{\rightarrow}$.

9.3. DEFINITION. Let $\boldsymbol{R}$ be a reduction relation.
(i) An *$\boldsymbol{R}$-reduction path* is a finite or infinite sequence

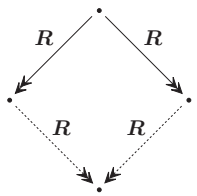
$$g_0 \underset{\boldsymbol{R}}{\rightarrow} g_1 \underset{\boldsymbol{R}}{\rightarrow} g_2 \underset{\boldsymbol{R}}{\rightarrow} \cdots .$$

(ii) A graph g is in *$\boldsymbol{R}$-normal form* ($\boldsymbol{R}$-nf) if there is no g' such that $g \underset{\boldsymbol{R}}{\rightarrow} g'$. The set of graphs in $\boldsymbol{R}$-normal form is denoted as $\mathrm{NF}_{\boldsymbol{R}}$.
(iii) *g has g' as $\boldsymbol{R}$-normal form* if $g \underset{\boldsymbol{R}}{\twoheadrightarrow} g'$ and $g' \in \mathrm{NF}_{\boldsymbol{R}}$.

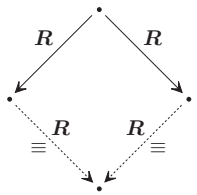
9.4. DEFINITION. Let $\boldsymbol{R}$ be a reduction relation.
(i) $\boldsymbol{R}$ is *Church-Rosser* (CR) or *confluent* if for all $g, g_1, g_2 \in \mathcal{G}$

$$g \underset{\boldsymbol{R}}{\twoheadrightarrow} g_1 \ \& \ g \underset{\boldsymbol{R}}{\twoheadrightarrow} g_2 \ \Rightarrow \ \exists h \, [g_1 \underset{\boldsymbol{R}}{\twoheadrightarrow} h \ \& \ g_2 \underset{\boldsymbol{R}}{\twoheadrightarrow} h].$$

In a picture:



(ii) $\boldsymbol{R}$ is *subcommutative* if

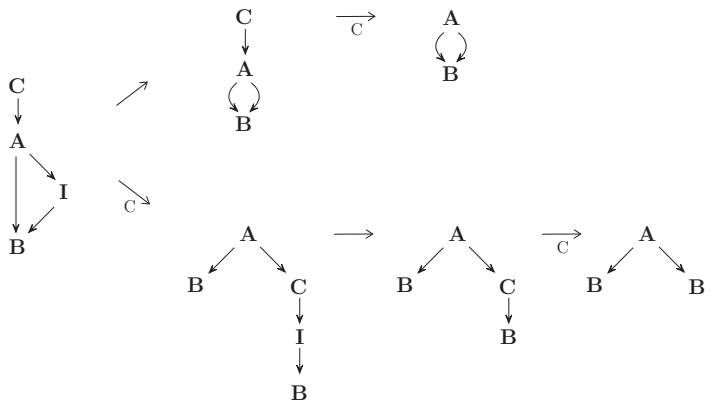


(iii) *g strongly $\boldsymbol{R}$-normalizes* (notation $\mathrm{SN}_{\boldsymbol{R}}(g)$) if there exists no infinite $\boldsymbol{R}$-reduction path starting with g.
(iv) $\boldsymbol{R}$ is *strongly normalizing* (SN) if for all g one has $\mathrm{SN}_{\boldsymbol{R}}(g)$.

Let $\mathcal{T} = \langle \mathcal{G}, \mathcal{R} \rangle$ a orthogonal cycle-free C-TGRS. We want to obtain a Church-Rosser-like result for $\mathcal{T}$.

A pure Church-Rosser result for $\mathcal{R}$C-reduction is impossible, as is shown in the following example.

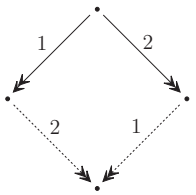
9.5. Example. Let $\mathbf{I}(x) \to x$ be the usual rule for the identity function. Then one has two 'divergent' reduction paths.



For this reason we study confluence of the system in which the copy action is decomposed. Our aim is to show that $\mathcal{R}_{\preceq}+-$-reduction is confluent.

A useful tool to prove the confluence of a combined reduction relation is the following result due to Hindley (1964) and Rosen (1973).

9.6. Definition. Let $\boldsymbol{R}_1, \boldsymbol{R}_2$ be reduction relations. Then $\boldsymbol{R}_1$ *commutes* with $\boldsymbol{R}_2$ if



9.7. Hindley-Rosen Lemma. *Let $\boldsymbol{R}_1, \boldsymbol{R}_2$ be reduction relations. Suppose*

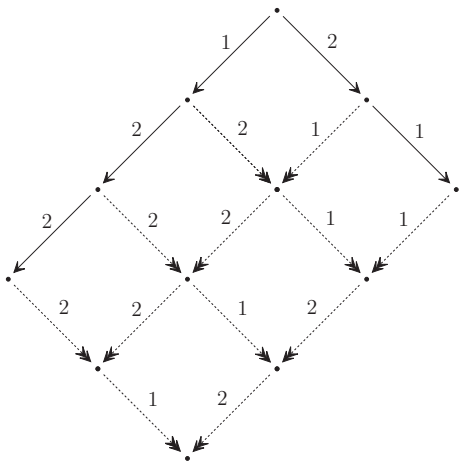
(1) *$\boldsymbol{R}_1$ is CR;*

(2) *$\boldsymbol{R}_2$ is CR;*

(3) *$\boldsymbol{R}_1$ commutes with $\boldsymbol{R}_2$.*

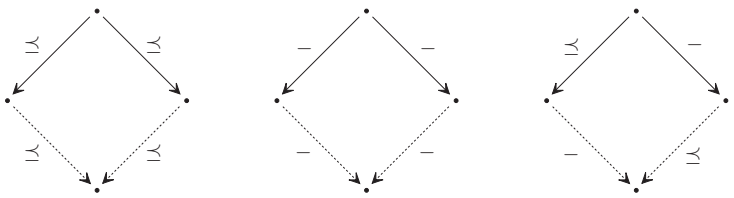
Then $\boldsymbol{R}_1\boldsymbol{R}_2$ is CR.

Proof. By a diagram chase suggested by the following picture.



□

9.8. Lemma. (i) (ii) (iii)

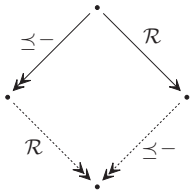


Proof. (i) By theorem 3.25.
(ii) By proposition 8.6.
(iii) By proposition 8.16. □

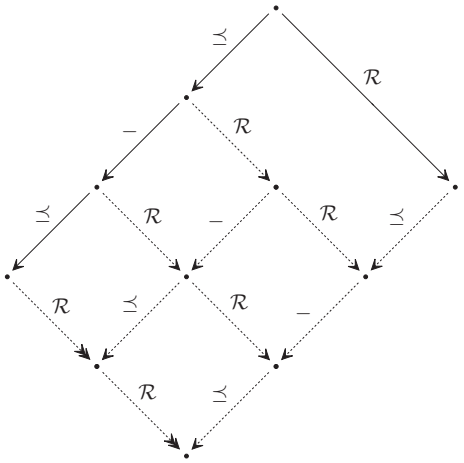
9.9. Proposition. $\preceq -$ *is CR.*

Proof. By the Hindley-Rosen lemma, using the facts that $\underset{\preceq}{\to}$ and $\underset{-}{\to}$ are Church-Rosser (lemma 9.8 (i, ii)), and that $\underset{\preceq}{\to}$ and $\underset{-}{\to}$ commute (lemma 9.8 (iii)). □

9.10. Lemma.



Proof. By propositions 6.6 and 8.14 and a diagram chase as indicated below.



□

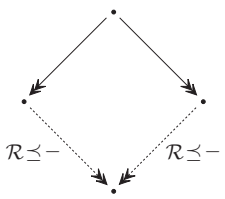
9.11. Theorem. $\mathcal{R}$ *is CR.*

Proof. $\mathcal{R}$-reduction is subcommutative by proposition 5.9. From this the confluence of $\mathcal{R}$ follows easily. □

9.12. Theorem. $\mathcal{R}\preceq -$ *is CR.*

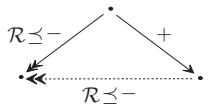
Proof. Again by the Hindley-Rosen lemma, using proposition 9.9, lemma 9.10 and theorem 9.11. □

Our next aim is to show the following. Below $\to$ stands for $\mathcal{R}\preceq+-$-reduction.

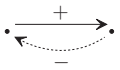


The proof is carried out in two main steps. First it is shown how +-reduction can be dealth with, and next the CR result for $\mathcal{R}\preceq-$-reduction is used.

9.13. Neutralization Lemma.

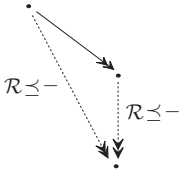


Proof. By the fact that $\underset{-}{\to}$ is able to neutralize $\underset{+}{\to}$:

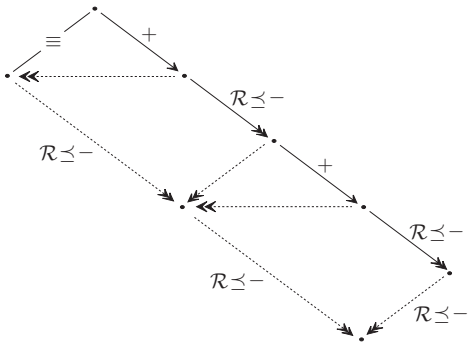


□

9.14. Proposition.

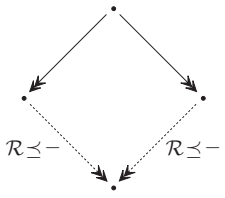


Proof. Using the neutralization lemma (triangles) and theorem 9.12 (quadrangles) one can erect the following diagram.

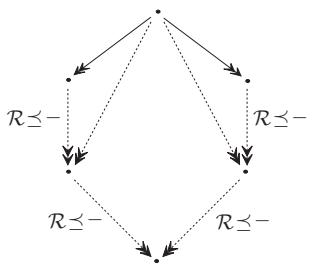


□

9.15. Theorem.

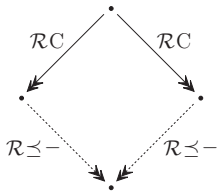


PROOF. By proposition 9.14 and theorem 9.12.



□

9.16. COROLLARY.



The relations between various normal forms can now be expressed.

9.17. DEFINITION. Let g, g' be graphs.

(i) g and g' are *copy tree equivalent* (notation $g \sim_{\mathrm{CT}} g'$) if they are tree equivalent after elimination of all present copy nodes, i.e.

$$\mathrm{E}_{\mathrm{C}(g)}(g) \sim_{\mathrm{T}} \mathrm{E}_{\mathrm{C}(g')}(g').$$

(ii) $\sim_{\preceq -}$ is the reflexive symmetric transitive closure of the reduction relation $\preceq -$.

9.18. THEOREM. *Let $g, g' \in \mathbb{G}$. Suppose $\mathrm{C}(g), \mathrm{C}(g')$ are cycle-free. Then*

$$g \sim_{\mathrm{CT}} g' \;\Leftrightarrow\; g \sim_{\preceq -} g'.$$

PROOF. ($\Rightarrow$) Using $\mathrm{U}(\mathrm{GC}(h)) = \mathrm{U}(h)$ one derives

$$\begin{array}{lll} g \sim_{\mathrm{CT}} g' & \Rightarrow & \mathrm{E}_{\mathrm{C}(g)}(g) \sim_{\mathrm{T}} \mathrm{E}_{\mathrm{C}(g')}(g') \\ & \Rightarrow & \mathrm{GC}(\mathrm{E}_{\mathrm{C}(g)}(g)) \sim_{\mathrm{T}} \mathrm{GC}(\mathrm{E}_{\mathrm{C}(g')}(g')) \\ & \Rightarrow & \mathrm{GC}(\mathrm{E}_{\mathrm{C}(g)}(g)) \uparrow \mathrm{GC}(\mathrm{E}_{\mathrm{C}(g')}(g')), \quad \text{by proposition 3.23.} \end{array}$$

($\Leftarrow$) By induction on the generation of $\sim_{\preceq -}$. The crucial case is $g \underset{\preceq}{\rightarrow} g'$. Then by lemma 9.8 (i) one has $\mathrm{GC}(\mathrm{E}_{\mathrm{C}(g)}(g)) \preceq \mathrm{GC}(\mathrm{E}_{\mathrm{C}(g')}(g'))$ and therefore $\mathrm{E}_{\mathrm{C}(g)}(g) \sim_{\mathrm{T}} \mathrm{E}_{\mathrm{C}(g')}(g')$. □

Normal forms turn out to be unique up to copy tree equivalence.

9.19. THEOREM. (i) *If g has g_1, g_2 as $\mathcal{R}$C-normal forms, then $g_1 \uparrow g_2$.*

(ii) *If g has g_1, g_2 as $\mathcal{R}$-normal forms, then $g_1 \sim_{\mathrm{CT}} g_2$.*

PROOF. (i) A graph in $\mathcal{R}$C-normal form is also in $\mathcal{R}-$-normal form. Now by theorem 9.15 we are done.

(ii) Note that $\preceq -$-reduction does not create new $\mathcal{R}$-redexes. Hence by theorem 9.15 and theorem 9.18 one has $g_1 \sim_{\mathrm{CT}} g_2$. □

10. Towards an implementation of lazy copying

If one uses term rewriting systems or term graph rewriting systems as a model of computation, these systems are usually equipped with a reduction strategy.

Let $\mathcal{T} = \langle \mathcal{G}, \mathcal{R} \rangle$ be a TGRS or C-TGRS. During graph reduction in $\mathcal{T}$ there may be several rewrite possibilities, since a graph may contain several redexes. A reduction *strategy* is a function that selects one or more reduction possibilities.

Firstly, the notion of strategy is explained for TGRS's. In the second part of this section, some aspects of strategies with respect to C-TGRS's are explained.

10.1. Definition. (i) A *reduction strategy* (for $\mathcal{R}$) is a function σ that takes each $g \in \mathcal{G}$ to a set of finite reduction paths with redex specifications starting with g. Thus the elements of $\sigma(g)$ are of the form

$$g = g_0 \overset{\Delta_0}{\underset{\mathcal{R}}{\rightarrow}} g_1 \overset{\Delta_1}{\underset{\mathcal{R}}{\rightarrow}} \cdots \overset{\Delta_{n-1}}{\underset{\mathcal{R}}{\rightarrow}} g_n.$$

(ii) The reduction relation induced by σ (notation $\underset{\sigma}{\rightarrow}$) is defined by

$$g \underset{\sigma}{\rightarrow} g' \quad \Leftrightarrow \quad \sigma(g) \text{ contains a reduction path from } g \text{ to } g'.$$

Note that $\underset{\sigma}{\rightarrow} \subseteq \underset{\mathcal{R}}{\rightarrow}^+$, and g is in σ-nf iff $\sigma(g) = \emptyset$.

10.2. Definition. Let σ be a reduction strategy for $\mathcal{R}$.

(i) σ is a *one step strategy* if $\underset{\sigma}{\rightarrow} \subseteq \underset{\mathcal{R}}{\rightarrow}$.

(ii) σ is *deterministic* if for each g one has that $\sigma(g)$ is either a singleton or empty. Otherwise σ is called *nondeterministic*.

(iii) σ is *normalizing* if

(1) $\mathrm{NF}_\sigma \subseteq \mathrm{NF}_\mathcal{R}$,

(2) g has an $\mathcal{R}$-nf $\Rightarrow$ $\mathrm{SN}_\sigma(g)$.

One is usually interested in effective strategies, i.e. strategies that are computable. In order to define this, assume that we have some effective encoding of graphs, redexes, finite reduction paths, and finite sets of such, as natural numbers.

10.3. Definition. A strategy σ is *computable* if $\sigma(g)$ is finite for all g, and moreover σ is total recursive (after encoding).

Note that if σ is computable, then the predicate 'is in σ-nf' is decidable (by the effectiveness of the encoding).

10.4. Definition. Let σ be a 1-step reduction strategy for $\mathcal{R}$.

(i) Let $g \in \mathcal{G}$. The set of *σ-redexes* of g (notation $\delta_\sigma(g)$) is the collection of redexes that are reduced in $\sigma(g)$, i.e.

$$\delta_\sigma(g) = \{\Delta \mid g \overset{\Delta}{\underset{\mathcal{R}}{\rightarrow}} g' \in \sigma(g) \quad \text{for some } g'\}.$$

(ii) σ is *nonambiguous* if for each g the elements of $\delta_\sigma(g)$ are pairwise disjoint.

For C-TGRS's the situation is a bit more complicated. As was already remarked in section 7, copying a graph may involve duplication of work (if the graph contains redexes).

If one wants to use graph rewriting with lazy copying in practice it is necessary to have some control over the choice of the segments that are duplicated during C-reduction.

We therefore extend the language of graph rewriting with the possibility of attaching a so called *defer* attribute to nodes. Intuitively, this attribute temporarily prevents the node in question of being copied. Defer attributes are also allowed to appear in the right-hand side of a rewrite rule. In Eekelen et al. (1991) these attributes are used to define primitives for indicating parallelism.

During graph rewriting, defer attributes are *added* and *propagated* while performing the graph operations extension, redirection and garbage collection, according to the following rules.

- If n is deferred in g, then n is deferred in $g + E$.
- If n is deferred in E, then n^* is deferred in $g + E$.
  (This applies e.g. to $g + \Delta$ for redexes $\Delta = \langle R, \mu \rangle$ in which $R \,|\, r$ contains deferred nodes.)
- If n is deferred in g, then $\rho(n)$ is deferred in $\rho(g)$.
- If n is deferred in g, and $r_g \rightsquigarrow n$, then n is deferred in $\mathrm{GC}(g)$.

The use of this attribute can be combined with a so called *pre-strategy* which selects either ordinary $\mathcal{R}$-redexes or copy nodes. The term pre-strategy is used here because C-redexes are specified only partially, by indicating just copy nodes. Therefore a pre-strategy can also be viewed as an ambiguous strategy for $\mathcal{R}$C-reduction.

It will now be described how such a pre-strategy can be used to obtain a 1-step strategy for $\mathcal{R}$C-reduction.

10.5. Definition. (i) $\mathrm{D}(g)$ denotes the collection of deferred nodes of g.

(ii) A *1-step pre-strategy* (for $\mathcal{T}$) is a function σ which takes each $g \in \mathcal{G}$ to a set of $\mathcal{R}$-redex roots, and/or copy nodes in g, such that for all $n \in \sigma(g)$

$$n \in \mathrm{C}(g) \;\Rightarrow\; succ(n) \notin \mathrm{D}(g),$$

i.e. σ will not attempt to copy a 'deferred graph'.

(iii) A graph is in *σ-normal form* if $\sigma(g) = \emptyset$.

(iv) σ is *deterministic* if $\sigma(g)$ is either a singleton or empty.

Now let σ be a pre-strategy for $\mathcal{T}$. Actual rewriting according to σ proceeds as follows. If σ selects an $\mathcal{R}$-redex root, the corresponding redex is contracted. Moreover, if σ points out a copy node then a copy reduction step is done. The subgraph $g \,|\, succ(n)$ is copied up to the deferred nodes of g. Thus

$$\begin{aligned} g \xrightarrow[\sigma]{} g' \quad \text{if} \quad & \text{for some } n \in \sigma(g) \\ & n \in \mathrm{C}(g), \text{ and } g \overset{\langle n,s \rangle}{\underset{\mathrm{C}}{\rightarrow}} g', \\ & \text{where } s = (g \mid succ(n)) \rtimes \mathrm{D}(g), \qquad \text{or} \\ & n \notin \mathrm{C}(g), \text{ and } g \overset{\Delta_n}{\underset{\mathcal{R}}{\rightarrow}} g', \\ & \text{where } \Delta_n \text{ is the } \mathcal{R}\text{-redex associated with } n. \end{aligned}$$

Moreover, after a rewrite step is performed, the defer attribute of a node $n \in g$ is *removed* if $g \mid n$ is in σ-nf. As a consequence, each graph in $\mathcal{R}$C-normal form does not have any deferred nodes.

Note that the above s is nonempty since $succ(n)$ is not deferred.

The concept of lazy copying has been incorporated in the programming language CONCURRENT CLEAN (see Nöcker et al. (1991)). In the implementation of this language the abovementioned technique is used in combination with a parallel version of the so called functional reduction strategy.

11. Conclusion

We have shown that term graph rewriting can be extended with lazy copying in an elegant way. The Church-Rosser theorem implies that the resulting reduction system is consistent in some sense, since normal forms are unique up to unraveling.

The following topics will be subject to further research. Firstly, an analogue of theorem 9.19 concerning graphs (not necessarily having a normal form) that have an initial part in normal form is desirable. Furthermore the effects of admitting cyclic graphs in TGRS's and C-TGRS's on our confluence results will be investigated. Moreover the notion of strategy (again for both kinds of systems) is to be studied.

References

Barendregt, H.P. (1984). *The Lambda Calculus: its syntax and semantics*, Studies in Logic 103, North-Holland, Amsterdam.

Barendregt, H.P., M.C.J.D. van Eekelen, J.R.W. Glauert, J.R. Kennaway, M.J. Plasmeijer and M.R. Sleep (1987a). Towards an intermediate language based on graph rewriting, *Proc. of Parallel Architectures and Languages Europe (PARLE)*, Eindhoven, The Netherlands, Springer Verlag, LNCS 259 II, pp. 159–175.

Barendregt, H.P., M.C.J.D. van Eekelen, J.R.W. Glauert, J.R. Kennaway, M.J. Plasmeijer and M.R. Sleep (1987b). Term graph reduction, *Proc. of Parallel Architectures and Languages Europe (PARLE)*, Eindhoven, The Netherlands, Springer Verlag, LNCS 259 II, pp. 141–158.

Brus, T., M.C.J.D. van Eekelen, M. van Leer, M.J. Plasmeijer and H.P. Barendregt (1987). Clean - a language for functional graph rewriting, *Proc. of Conference on Functional Programming Languages and Computer Architecture (FPCA '87)*, Portland, Oregon, USA, Springer Verlag, LNCS 274, pp. 364–384.

Eekelen, M.C.J.D. van, M.J. Plasmeijer and J.E.W. Smetsers (1991). Parallel graph rewriting on loosely coupled machine architectures, *in:* Kaplan and Okada (eds.), *Proc. of Conditional and Typed Rewriting Systems (CTRS'90)*, Montreal, Canada, Springer Verlag, LNCS 516, pp. 354–369.

Hindley, J.R. (1964). The church-rosser property and a result in combinatory logic, *Desertation*, University of Newcastle-upon-Tyne.

Klop, J.W. (1992). Term rewriting systems, *in:* Gabbay Abramsky and Maibaum (eds.), *Handbook of Logic in Computer Science*, Oxford University Press, I.

Lamping, J. (1990). An algorithm for optimal lambda calculus reduction,, *Proc. of POPL'90: Seventeenth annual ACM symposium on Principles of Programming Languages*, San Francisco, California, pp. 16–30.

Nöcker, E.G.J.M.H., J.E.W. Smetsers, M.C.J.D. van Eekelen and M.J. Plasmeijer (1991). Concurrent clean, *Proc. of Parallel Architectures and Languages Europe (PARLE'91)*, Eindhoven, The Netherlands, Springer Verlag, LNCS 505, pp. 202–219.

Rosen, B.K. (1973). Tree manipulation systems and church-rosser theorems, *J. Assoc. Comput. Mach.*.

Smetsers, J.E.W., E.G.J.M.H. Nöcker, J.H.G. van Groningen and M.J. Plasmeijer (1991). Generating efficient code for lazy functional languages, *Proc. of Conference on Functional Programming Languages and Computer Architecture (FPCA '91)*, Cambridge, MA, USA, Springer Verlag, LNCS 523, pp. 592–617.

Wadsworth, C.P. (1971). *Semantics and Pragmatics of the Lambda Calculus*, Dissertation, Oxford University.