Electronic Notes in Theoretical Computer Science 51 (2001)
URL: http://www.elsevier.nl/locate/entcs/volume51.html 13 pages

Essentials of Term Graph Rewriting

Detlef Plump

Department of Computer Science, The University of York
Heslington, York YO10 5DD, United Kingdom
det@cs.york.ac.uk

Abstract

Term graph rewriting is a model for computing with graphs representing functional expressions. Graphs allow to share common subexpressions which improves the efficiency of conventional term rewriting in space and time. This article reviews essentials of term graph rewriting concerning soundness, completeness, termination and confluence.

1 Introduction

Term graph rewriting is about the representation of functional expressions as graphs and their evaluation by rule-based graph transformation. The motivation for using graphs is to improve the efficiency of conventional term rewriting which is based on strings or trees. For example, consider the following term rewrite rules defining the factorial function on natural numbers (where $\mathtt{s}$ denotes the successor function):

$$\mathtt{fact}(0) \to \mathtt{s}(0)$$
$$\mathtt{fact}(\mathtt{s}(\mathtt{x})) \to \mathtt{s}(\mathtt{x}) \times \mathtt{fact}(\mathtt{x})$$

Evaluating a term [1] $\mathtt{fact}(\mathtt{s}^n(0))$ by these two rules yields

$$\mathtt{s}^n(0) \times (\mathtt{s}^{n-1}(0) \times (\ldots \mathtt{s}^2(0) \times (\mathtt{s}(0) \times \mathtt{s}(0)) \ldots)),$$

which is a term containing $\sum_{i=2}^{n+2} i = (n^2+5n+4)/2$ function symbols. Thus the space needed by this evaluation is quadratic in the size of the input term. The non-linear behaviour is caused by the second rule which duplicates the subterm matched by the variable $\mathtt{x}$. In general, every term rewrite rule containing some variable more often on its right-hand side than on its left-hand side can increase the size of a term by a non-constant amount.

In implementations this problem is usually overcome by creating several pointers to a subterm instead of copying it. This *sharing* of common subterms

[1] Here $\mathtt{s}^n(0)$ stands for the term $\mathtt{s}(\mathtt{s}(\ldots \mathtt{s}(0) \ldots))$ with n occurrences of $\mathtt{s}$.

means that one moves from the computational model of term rewriting to that of term graph rewriting. For example, Figure 1 shows the evaluation of $\mathtt{fact}(\mathtt{s}^n(0))$ by term graph rewriting with the above rules. This evaluation needs only space $2n+3$ and hence is linear in the size of the input term.[2]

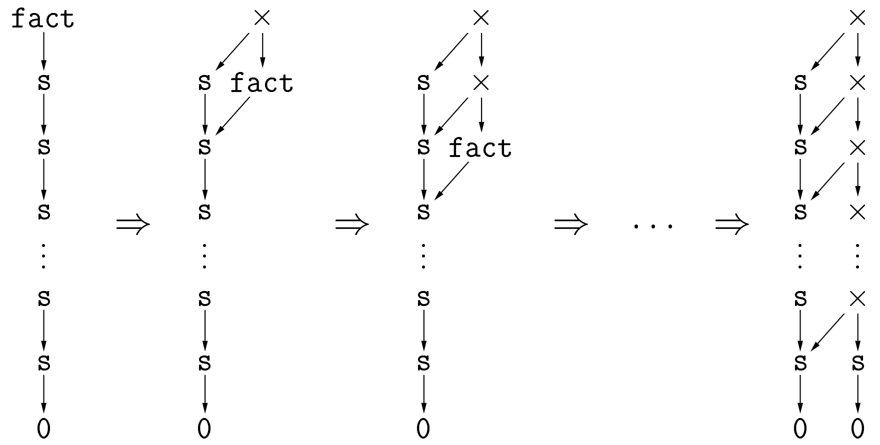

Fig. 1. Evaluating $\mathtt{fact}(\mathtt{s}^n(0))$ by term graph rewriting

Employing term graph rewriting instead of term rewriting—or *implementing* the latter by the former, if you like—has consequences beyond improving efficiency. Sharing common subterms prevents certain term rewrite sequences, namely those where equal subterms are rewritten differently. Hence, a priori, it is not clear in what sense (if at all) term graph rewriting is a sound and complete implementation of term rewriting. For the same reason it is unclear whether relevant properties of rewrite system, notably termination and confluence, are preserved when switching between the two models of computation.

The following sections review essentials of term graph rewriting, viz. soundness with respect to term rewriting, completeness for equational proofs, and the relation to term rewriting concerning termination and confluence. The presentation is oriented by the survey [15].

2 Term graph rewriting

Let Σ and X be disjoint sets of *function symbols* and *variables*, respectively, where each function symbol f comes with a natural number $\text{arity}(f) \geq 0$ and variables have arity 0. Function symbols of arity 0 are called *constants*.

A *hypergraph* over Σ and X is a system $G = \langle \mathrm{V}_G, \mathrm{E}_G, \mathrm{lab}_G, \mathrm{att}_G \rangle$ consisting of two finite sets V_G and E_G of *nodes* and *hyperedges*, a labelling function

[2] Note also that in this example both term rewriting and term graph rewriting need $n+1$ rule applications to reach a result, but single term rewrite steps (at given positions in terms) require more than constant time since they have to copy subterms of non-constant size.

$\mathrm{lab}_G\colon \mathrm{E}_G \to \Sigma \cup \mathrm{X}$, and an attachment function $\mathrm{att}_G\colon \mathrm{E}_G \to \mathrm{V}_G^*$ which assigns a string of nodes to a hyperedge e such that the length of $\mathrm{att}_G(e)$ is $1 + \mathrm{arity}(\mathrm{lab}_G(e))$. From now on hypergraphs and hyperedges are simply called graphs and edges.

Given a graph G and an edge e with $\mathrm{att}_G(e) = v\, v_1 \ldots v_n$, node v is the *result node* of e while $v_1, \ldots, v_n$ are the *argument nodes*. The result node v is denoted by $\mathrm{res}(e)$. For each node v, $G[v]$ is the subgraph consisting of all nodes that are reachable from v and all edges having these nodes as result nodes.

Definition 2.1 (Term graph) A graph G is a *term graph* if

(1) there is a node root_G from which each node is reachable,

(2) G is acyclic, and

(3) each node is the result node of a unique edge.

Figure 2 shows three term graphs with binary function symbols `f`, `g` and `h`, and a constant `a`. Edges are depicted as boxes with inscribed labels, and bullets represent nodes. A line connects each edge with its result node, while arrows point to the argument nodes. The order in the argument string is given by the left-to-right order of the arrows leaving the box. This graphical format for term graphs is in bijective correspondence with the more compact format used in the introduction. In the sequel both versions will be used.

A *term* over Σ and X is a variable, a constant, or a string $f(t_1, \ldots, t_n)$ where f is a function symbol of arity $n \geq 1$ and $t_1, \ldots, t_n$ are terms.

Definition 2.2 (Term representation) Let v be a node in a term graph G, e be the unique edge with result node v, and $\mathrm{att}_G(e) = v\, v_1 \ldots v_n$. Then

$$\mathrm{term}_G(v) = \begin{cases} \mathrm{lab}_G(e) & \text{if } n = 0, \\ \mathrm{lab}_G(e)(\mathrm{term}_G(v_1), \ldots, \mathrm{term}_G(v_n)) & \text{otherwise} \end{cases}$$

is the term represented by v. Let $\mathrm{term}(G)$ be an abbreviation for $\mathrm{term}_G(\mathrm{root}_G)$.

A *graph morphism* $f\colon G \to H$ between two graphs G and H consists of two functions $f_\mathrm{V}\colon \mathrm{V}_G \to \mathrm{V}_H$ and $f_\mathrm{E}\colon \mathrm{E}_G \to \mathrm{E}_H$ that preserve labels and attachment to nodes, that is, $\mathrm{lab}_H \circ f_\mathrm{E} = \mathrm{lab}_G$ and $\mathrm{att}_H \circ f_\mathrm{E} = f_\mathrm{V}^* \circ \mathrm{att}_G$ (where $f_\mathrm{V}^*\colon \mathrm{V}_G^* \to \mathrm{V}_H^*$ maps a string $v_1 \ldots v_n$ to $f_\mathrm{V}(v_1) \ldots f_\mathrm{V}(v_n)$). The morphism f is *injective* (*surjective*) if f_V and f_E are. If f is injective and surjective, then it is an *isomorphism*. In this case G and H are *isomorphic*, which is denoted by $G \cong H$. For technical convenience, isomorphic term graphs will be considered as equal in this article. (This convention is formally justified in [15].)

Definition 2.3 (Collapsing) Given two term graphs G and H, G *collapses* to H if there is a graph morphism $G \to H$ mapping root_G to root_H. This is denoted by $G \succeq H$ or, if the morphism is non-injective, by $G \succ H$. The latter kind of collapsing is said to be *proper*. A term graph G is *fully collapsed* if there is no H with $G \succ H$, while G is a *tree* if there is no H with $H \succ G$.

Figure 2 shows two collapse steps where the middle graph is fully collapsed. For every term graph G there are a tree $\triangle G$ and a fully collapsed term graph ∇G such that $\mathrm{term}(G) = \mathrm{term}(\triangle G) = \mathrm{term}(\nabla G)$, where $\triangle G$ and ∇G are unique up to isomorphism [12].

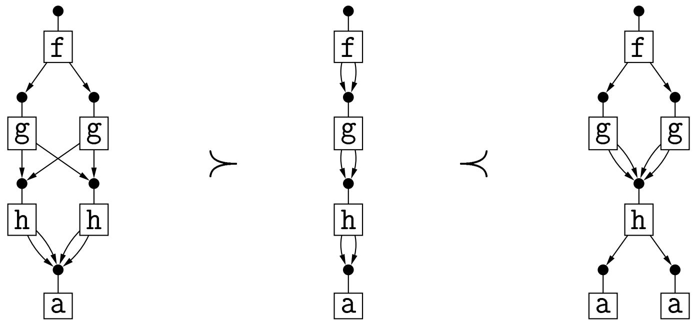

Fig. 2. Two collapse steps

A *term rewrite rule* $l \to r$ consists of two terms l and r over Σ and X such that l is not a variable and all variables in r occur also in l. A set $\mathcal{R}$ of term rewrite rules is a *term rewriting system.* The reader is assumed to be familiar with basic concepts of term rewriting (see [2]). In the following $\mathcal{R}$ denotes an arbitrary term rewiting system. The term rewrite relation associated with $\mathcal{R}$ is denoted by $\to$, its n-fold composition by $\to^n$, its transitive closure by $\to^+$, its reflexive-transitive closure by $\to^*$, and its reflexive-symmetric-transitive closure by $\leftrightarrow^*$.

Given a term t, denote by $\Diamond t$ a term graph representing t such that (1) each node v with an indegree greater than one satisfies $\mathrm{term}_{\Diamond t}(v) \in \mathrm{X}$ and (2) for each variable x in t there is a unique node v with $\mathrm{term}_{\Diamond t}(v) = x$. Thus $\Diamond t$ is obtained from $\triangle t$ by merging edges labelled with equal variables. The graph resulting from $\Diamond t$ after removing all edges labelled with variables is denoted by $\underline{\Diamond t}$. A term graph G is an *instance* of $\Diamond t$ if there is graph morphism $\underline{\Diamond t} \to G$ sending $\mathrm{root}_{\Diamond t}$ to root_G.

Definition 2.4 (Term graph rewriting) Let G and H be term graphs, $l \to r$ be a rewrite rule in $\mathcal{R}$, and v be a node in G such that $G[v]$ is an instance of $\Diamond l$. Then there is a *proper rewrite step* $G \Rightarrow_{v, l \to r} H$ if H is obtained from G by the following construction:

(1) Remove the unique edge whose result node is v, yielding a graph G_1.

(2) Construct a graph G_2 from the disjoint union $G_1 + \underline{\Diamond r}$ by
- merging v with $\mathrm{root}_{\Diamond r}$, and
- merging the image of u with u', for all nodes u in $\Diamond l$ and u' in $\Diamond r$ such that $\mathrm{term}_{\Diamond l}(u) = \mathrm{term}_{\Diamond r}(u') \in \mathrm{X}$.

(3) Remove from G_2 all nodes and edges that are not reachable from root_G ("garbage collection"), yielding $H = G_2[\mathrm{root}_G]$.

Such a rewrite step is also denoted by $G \Rightarrow_v H$ or just $G \Rightarrow H$, and $\Rightarrow^*$ is the reflexive-transitive closure of $\Rightarrow$.

The combined effect of steps (1) and (2) can also be specified through a rule in the so-called double-pushout approach to graph transformation. By translating term rewrite rules into suitable graph transformation rules one obtains a model of term graph rewriting without garbage collection which is known as *jungle evaluation* [5,6]. Details can be found in [12], where term graph rewriting is defined by extending jungle evaluation with garbage collection.

A technically somewhat different approach for modelling term rewriting by graph rewriting was introduced in [3], where the name "term graph rewriting" was coined. The definition of rewrite steps in that paper involves the redirection of edges and garbage collection, and is in its effect (on acyclic graphs) equivalent to Definition 2.4.

3 Soundness and completeness

It turns out that term graph rewriting is sound with respect to term rewriting in that every step $G \Rightarrow_{v,\, l \to r} H$ corresponds to a sequence of applications—or a parallel application—of the rule $l \to r$ to occurrences of the subterm $\mathrm{term}_G(v)$ in $\mathrm{term}(G)$.

Theorem 3.1 (Soundness [7]) *For all term graphs G and H,*

$$G \underset{v}{\Rightarrow} H \text{ implies } \mathrm{term}(G) \xrightarrow{n} \mathrm{term}(H)$$

where n is the number of paths from root_G to v.

This result also explains the potential speed-up in the number of rewrite steps when using term graph rewriting instead of term rewriting. Moreover, it follows immediately that for every sequence of term graph rewrite steps there is a corresponding term rewrite sequence of equal or greater length. Hence, if the worst-case time complexity of a term or term graph is approximated by the maximal length of rewrite sequences starting from it [3], the worst-case time behaviour of term graph rewriting is never worse than that of term rewriting.

Analogously, the space needed by term graph rewriting in the worst-case cannot exceed the space needed by term rewriting. This is because the graphs of a rewrite sequence are at most as large as the terms they represent and which occur in the corresponding term rewrite sequence.

Having considered the soundness of term graph rewriting, a natural question is whether it is also complete. The answer depends on the form of com-

[3] This measure is an approximation because it neglects both the time needed for finding positions in terms and graphs at which rules can be applied and the additional time needed for term rewrite steps with rules containing repeated variables in their right-hand sides.

pleteness we are looking for. If completeness is regarded as the ability to simulate every term rewrite sequence, then a simple counterexample is given by the rules

$$\texttt{f}(\texttt{x}) \to \texttt{g}(\texttt{x}, \texttt{x})$$

$$\texttt{a} \to \texttt{b}$$

and the rewrite sequence $\texttt{f}(\texttt{a}) \to \texttt{g}(\texttt{a}, \texttt{a}) \to \texttt{g}(\texttt{a}, \texttt{b})$. Starting from the tree $\triangle\texttt{f}(\texttt{a})$, it is impossible to reach $\triangle\texttt{g}(\texttt{a}, \texttt{b})$ by term graph rewriting because of the sharing introduced by the first rule. The only graphs derivable from $\triangle\texttt{f}(\texttt{a})$ are $\nabla\texttt{g}(\texttt{a}, \texttt{a})$, $\nabla\texttt{g}(\texttt{b}, \texttt{b})$ and $\triangle\texttt{f}(\texttt{b})$. To overcome this problem one can add copy steps to the rewrite relation, where *copying* is the inverse relation to collapsing. (See [1] for some results on term graph rewriting with copying.) However, copying is in conflict with the idea behind term graph rewriting to improve efficiency by sharing. The approach used below is rather to add proper collapse steps to the rewrite relation, which will make term graph rewriting complete for equational reasoning.

A second obstacle to the completeness of term graph rewriting are term rewrite rules with repeated variables in their left-hand sides. For instance, the rule $\texttt{eq}(\texttt{x}, \texttt{x}) \to \texttt{true}$ cannot be applied to the tree $\triangle\texttt{eq}(\texttt{0}, \texttt{0})$ because there is no graph morphism $\underline{\Diamond\texttt{eq(x,x)}} \to \triangle\texttt{eq}(\texttt{0}, \texttt{0})$ (see Figure 3). Hence, $\triangle\texttt{eq}(\texttt{0}, \texttt{0})$ is not reducible by $\Rightarrow$ although the term $\texttt{eq}(\texttt{0}, \texttt{0})$ is reducible. Figure 3 demonstrates that this problem can be solved by collapsing: merging the two edges labelled with 0 makes the rewrite rule applicable.

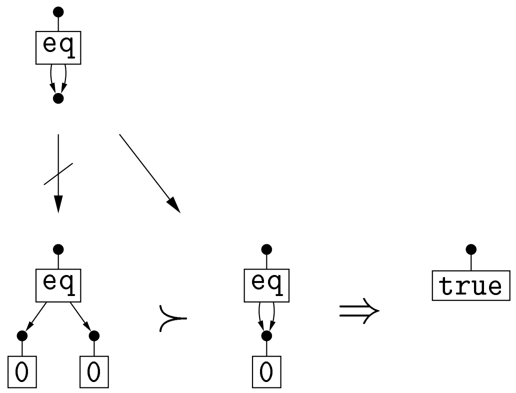

Fig. 3. Collapsing to enable a rule application

Definition 3.2 ($\Rightarrow_{\text{coll}}$) The relation $\Rightarrow_{\text{coll}}$ on term graphs is defined by

$$\underset{\text{coll}}{\Rightarrow} \;=\; \Rightarrow \cup \succ .$$

So rewriting by $\Rightarrow_{\text{coll}}$ allows proper rewriting and collapsing in an arbitrary order. It turns out that this relation is complete for a central application of term rewriting, namely equational reasoning. Considering the rules of $\mathcal{R}$ as equations $l \approx r$, the *equational theory* of $\mathcal{R}$ is the set of all equations $t \approx u$ such that $t \leftrightarrow^* u$. By Birkhoff's Theorem the equational theory coincides with the set of equations that are valid in all models of $\mathcal{R}$ (that is, in all algebras satisfying the equations in $\mathcal{R}$). Details can be found in [2].

Theorem 3.3 (Completeness [11]) *For all term graphs G and H,*

$$\mathrm{term}(G) \overset{*}{\leftrightarrow} \mathrm{term}(H) \text{ if and only if } G \underset{\mathrm{coll}}{\overset{*}{\Leftrightarrow}} H.$$

Here $\Leftrightarrow^*_{\mathrm{coll}}$ is the reflexive-symmetric-transitive closure of $\Rightarrow_{\mathrm{coll}}$. Thus, an equation $t \approx u$ is valid in the models of $\mathcal{R}$ if and only if there is a sequence of steps with $\Rightarrow_{\mathrm{coll}}$ and its inverse between two term graphs representing t and u. In other words, term graph rewriting with collapsing is complete for proving validity of equations in the same sense as term rewriting is. For example, consider again the system $\mathcal{R} = \{\mathrm{f(x)} \to \mathrm{g(x,x)},\ \mathrm{a} \to \mathrm{b}\}$. A proof of the validity of $\mathrm{f(a)} \approx \mathrm{g(a,b)}$ is given by

$$\triangle \mathtt{f(a)} \underset{\mathrm{coll}}{\Rightarrow} \nabla \mathtt{g(a,a)} \underset{\mathrm{coll}}{\Rightarrow} \nabla \mathtt{g(b,b)} \underset{\mathrm{coll}}{\Leftarrow} \triangle \mathtt{g(b,b)} \underset{\mathrm{coll}}{\Leftarrow} \triangle \mathtt{g(a,b)}.$$

Note that term graph rewriting without collapsing lacks the completeness of Theorem 3.3, for it is clear that $\triangle \mathtt{f(a)} \not\Leftrightarrow^* \triangle \mathtt{g(a,b)}$ in the above example.

The equational theory of $\mathcal{R}$ is *decidable* if there is a procedure that decides for arbitrary terms t and u whether $t \leftrightarrow^* u$ or not. There exist term rewriting systems whose equational theory is not decidable [2]. Theorem 3.3 suggests a decision procedure for the case that $\Rightarrow_{\mathrm{coll}}$ is convergent, improving the classical decision procedure based on term rewriting. Call a binary relation $\to$ on a given set *terminating* if there is no infinite sequence $a_1 \to a_2 \to \ldots$, and *confluent* if for every constellation $a_1 \leftarrow^* a \to^* a_2$ there is an element b such that $a_1 \to^* b \leftarrow^* a_2$. If $\to$ is both terminating and confluent, then it is *convergent*.

Corollary 3.4 *The equational theory of a finite term rewriting system is decidable if $\Rightarrow_{\mathrm{coll}}$ is convergent.*

For, if $\Rightarrow_{\mathrm{coll}}$ is confluent then for all term graphs G_1 and G_2 with $G_1 \Leftrightarrow^*_{\mathrm{coll}} G_2$ there is a term graph H such that $G_1 \Rightarrow^*_{\mathrm{coll}} H \Leftarrow^*_{\mathrm{coll}} G_2$. One represents the input terms t and u by term graphs, say ∇t and ∇u, and reduces these by $\Rightarrow_{\mathrm{coll}}$ to irreducible graphs. The latter is possible since $\Rightarrow_{\mathrm{coll}}$ is terminating. Now Theorem 3.3 implies that $t \leftrightarrow^* u$ if and only if the resulting irreducible graphs are equal.

This procedure is often more efficient than the corresponding procedure based on term rewriting. In addition it is more widely applicable because the class of convergent term rewriting systems is properly included in the class of systems for which $\Rightarrow_{\mathrm{coll}}$ is convergent. The latter will become apparent in the next two sections.

4 Termination

From Theorem 3.1 (soundness) and the fact that there are no infinite chains of proper collapse steps (since these decrease the size of graphs) it follows that term graph rewriting is terminating whenever term rewriting is.

Theorem 4.1 *If $\rightarrow$ is terminating, then $\Rightarrow_{\mathrm{coll}}$ is terminating as well.*

As a consequence, the comprehensive machinery developed for proving termination of term rewriting (see for example [2,4]) is applicable to term graph rewriting. But the following example demonstrates that term graph rewriting terminates in strictly more cases than term rewriting.

Term rewriting with the two rules

$$\mathtt{f}(\mathtt{a},\mathtt{b},\mathtt{x}) \rightarrow \mathtt{f}(\mathtt{x},\mathtt{x},\mathtt{x})$$

$$\mathtt{a} \rightarrow \mathtt{b}$$

is not terminating as there is the infinite rewrite sequence

$$\mathtt{f}(\mathtt{a},\mathtt{b},\mathtt{a}) \rightarrow \mathtt{f}(\mathtt{a},\mathtt{a},\mathtt{a}) \rightarrow \mathtt{f}(\mathtt{a},\mathtt{b},\mathtt{a}) \rightarrow \ldots$$

To see that $\Rightarrow_{\mathrm{coll}}$ is terminating, consider the following function τ from term graphs to natural numbers: For every term graph G, define $\tau(G) = m+n+p$, where m is the number of $\mathtt{f}$-labelled edges the first two argument nodes of which are distinct, n is the number of $\mathtt{a}$-labelled edges, and p is the number of nodes in G. One easily checks that $G \Rightarrow_{\mathrm{coll}} H$ implies $\tau(G) > \tau(H)$, so there cannot be an infinite sequence of $\Rightarrow_{\mathrm{coll}}$-steps.

The better termination behaviour of term graph rewriting raises the question whether there are general proof methods covering systems as the one above which is terminating under term graph rewriting but not under term rewriting. Such a method is the *recursive path order* on term graphs [14] which generalizes the well-known recursive path order for term rewriting systems [4] to the term graph setting. The recursive path order is not discussed here for lack of space.

Another termination proof technique that can cover non-terminating term rewriting systems is the combination of terminating systems. Toyama [16] observed that the union of two terminating term rewriting systems may not be terminating even if the systems have disjoint sets of function symbols. He gave the following example:

$$\mathcal{R}_0 \left\{ \mathtt{f}(\mathtt{0},\mathtt{1},\mathtt{x}) \rightarrow \mathtt{f}(\mathtt{x},\mathtt{x},\mathtt{x}) \right.$$

$$\mathcal{R}_1 \left\{ \begin{array}{l} \mathtt{g}(\mathtt{x},\mathtt{y}) \rightarrow \mathtt{x} \\ \mathtt{g}(\mathtt{x},\mathtt{y}) \rightarrow \mathtt{y} \end{array} \right.$$

It is not difficult to show that both systems are terminating, but their union admits the following infinite rewrite sequence:

$$\begin{aligned} \mathtt{f(g(0,1),g(0,1),g(0,1))} &\overset{2}{\to} \mathtt{f(0,1,g(0,1))} \\ &\to \mathtt{f(g(0,1),g(0,1),g(0,1))} \\ &\to \dots \end{aligned}$$

In contrast, termination of $\Rightarrow_{\mathrm{coll}}$ is preserved by the union of two systems even if they have certain function symbols in common. Call two term rewriting systems $\mathcal{R}_0$ and $\mathcal{R}_1$ *crosswise disjoint* if the function symbols in the left-hand sides of the rules in $\mathcal{R}_i$ do not occur in the right-hand sides of the rules in $\mathcal{R}_{1-i}$, for $i = 0, 1$. Also, for the following theorem, denote the relation $\Rightarrow_{\mathrm{coll}}$ over a term rewriting system $\mathcal{R}$ by $\Rightarrow_{\mathcal{R}}$.

Theorem 4.2 ([10]) *Let $\mathcal{R}_0 \cup \mathcal{R}_1$ be the union of two crosswise disjoint term rewriting systems. Then $\Rightarrow_{\mathcal{R}_0 \cup \mathcal{R}_1}$ is terminating if and only if $\Rightarrow_{\mathcal{R}_0}$ and $\Rightarrow_{\mathcal{R}_1}$ are terminating.*

Considering Toyama's example again, the attempt to simulate the cyclic rewrite sequence shown above yields the terminating rewrite sequences of Figure 4.

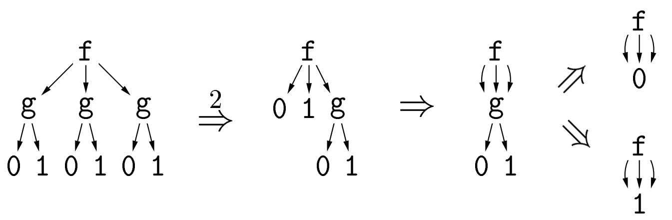

Fig. 4. Two terminating rewrite sequences

A generalization of Theorem 4.2 permitting even more common function symbols in $\mathcal{R}_0$ and $\mathcal{R}_1$ is presented in [9].

5 Confluence

The relation between term graph rewriting and term rewriting with respect to confluence is just opposite to the relation with respect to termination: the class of confluent term rewriting systems properly includes the class of systems over which term graph rewriting is confluent. The following theorem has a straightforward proof using the completeness of $\Rightarrow_{\mathrm{coll}}$ (Theorem 3.3).

Theorem 5.1 ([11]) *If $\Rightarrow_{\mathrm{coll}}$ is confluent, then $\to$ is confluent as well.*

The converse of this result is refuted by a counterexample from [11]:

$$\mathtt{f(x)} \to \mathtt{g(x,x)}$$
$$\mathtt{a} \to \mathtt{b}$$
$$\mathtt{g(a,b)} \to \mathtt{c}$$
$$\mathtt{g(b,b)} \to \mathtt{f(a)}$$

Using structural induction on terms one can show that every term reduces to a unique irreducible term, hence term rewriting is confluent. But Figure 5 shows that $\Rightarrow$ is not confluent, which clearly holds for $\Rightarrow_{\mathrm{coll}}$ as well. The problem here is that the sharing created by the rule $\mathtt{f(x)} \to \mathtt{g(x,x)}$ prevents the rewrite step $\mathtt{g(a,a)} \to \mathtt{g(a,b)}$ which is necessary to reduce $\mathtt{g(a,a)}$ to $\mathtt{c}$.

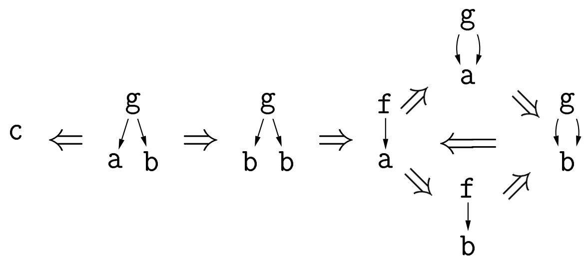

Fig. 5. Non-confluence of term graph rewriting

Despite the fact that term graph rewriting over a confluent term rewriting system needs not be confluent, no graph will reduce to more than one irreducible graph. Given a binary relation $\to$ on some set, call an element a a *normal form* if there is no b with $a \to b$. The relation $\to$ has *unique normal forms* if for all normal forms a and b, $a \leftrightarrow^* b$ implies $a = b$.

Theorem 5.2 ([12]) *The relation $\Rightarrow_{\mathrm{coll}}$ has unique normal forms if and only if $\to$ has unique normal forms.*

A binary relation $\to$ on some set is *normalizing* if for each element a there is a normal form b such that $a \to^* b$. Note that every terminating relation is normalizing, but the converse does not hold. Theorem 5.2 permits an easy proof of the following two properties.

Corollary 5.3 (1) *If $\Rightarrow_{\mathrm{coll}}$ is normalizing, then $\Rightarrow_{\mathrm{coll}}$ is confluent if and only if $\to$ is confluent.*
(2) *If $\to$ is convergent, then $\Rightarrow_{\mathrm{coll}}$ is convergent as well.*

The next example refutes the converse of Corollary 5.3(2), so the class of convergent term rewriting systems is properly included in the class of systems over which term graph rewriting is convergent. Consider the following system:

$$\mathtt{f(x)} \to \mathtt{g(x,x)}$$

$$\mathtt{a} \to \mathtt{b}$$

$$\mathtt{g(a,b)} \to \mathtt{f(a)}$$

Here every term has a unique normal form, so term rewriting is confluent. But the system is not terminating in view of the infinite rewrite sequence

$$\mathtt{f(a)} \to \mathtt{g(a,a)} \to \mathtt{g(a,b)} \to \mathtt{f(a)} \to \ldots$$

In contrast, by the above mentioned recursive path order on term graphs one can prove that term graph rewriting is terminating. Confluence follows then by Corollary 5.3(1).

Analogously to the situation for term rewriting [8], termination of term graph rewriting admits a decision procedure for confluence based on the analysis of so-called *critical pairs*. Details can be found in [12,13].

Theorem 5.4 ([12]) *There is an algorithm solving the following problem:*

Instance: *A finite term rewriting system $\mathcal{R}$ such that $\Rightarrow_{\mathrm{coll}}$ is terminating.*
Question: *Is $\Rightarrow_{\mathrm{coll}}$ confluent?*

Finally, confluence and convergence of term graph rewriting over combined systems will be considered. In contrast to the situation for term rewriting [17], unions of disjoint systems need not preserve confluence of $\Rightarrow_{\mathrm{coll}}$. In fact, confluence can get lost by merely extending the set Σ of function symbols [11].

Convergence of $\Rightarrow_{\mathrm{coll}}$, on the other hand, is preserved by the union of two crosswise disjoint systems if their left-hand sides do not mutually overlap. Call two term rewriting system $\mathcal{R}_0$ and $\mathcal{R}_1$ *non-interfering* if no left-hand side of $\mathcal{R}_i$ overlaps a left-hand side of $\mathcal{R}_{1-i}$, for $i = 0, 1$. (See [2] for a definition of "overlap".) As in the previous section, $\Rightarrow_{\mathcal{R}}$ denotes the relation $\Rightarrow_{\mathrm{coll}}$ over a term rewriting system $\mathcal{R}$.

Theorem 5.5 ([11]) *Let $\mathcal{R}_0 \cup \mathcal{R}_1$ be the union of two crosswise disjoint and non-interfering term rewriting systems. If $\Rightarrow_{\mathcal{R}_0}$ and $\Rightarrow_{\mathcal{R}_1}$ are convergent, then $\Rightarrow_{\mathcal{R}_0 \cup \mathcal{R}_1}$ is convergent as well.*

This result contrasts with the situation for term rewriting where even disjoint unions need not preserve convergence [16]. See also [9] for a generalization of Theorem 5.5 relaxing crosswise disjointness.

References

[1] Zena M. Ariola, Jan Willem Klop, and Detlef Plump. Bisimilarity in term graph rewriting. *Information and Computation*, 156(1/2):2–24, 2000.

[2] Franz Baader and Tobias Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[3] Henk Barendregt, Marko van Eekelen, John Glauert, Richard Kennaway, Rinus Plasmeijer, and Ronan Sleep. Term graph rewriting. In *Proc. Parallel Architectures and Languages Europe*, volume 259 of *Lecture Notes in Computer Science*, pages 141–158. Springer-Verlag, 1987.

[4] Nachum Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3:69–116, 1987.

[5] Annegret Habel, Hans-Jörg Kreowski, and Detlef Plump. Jungle evaluation. In *Proc. Recent Trends in Data Type Specification*, volume 332 of *Lecture Notes in Computer Science*, pages 92–112. Springer-Verlag, 1988.

[6] Berthold Hoffmann and Detlef Plump. Jungle evaluation for efficient term rewriting. In *Proc. Algebraic and Logic Programming.* Mathematical Research 49, pages 191–203, Berlin, 1988. Akademie-Verlag. Also in Springer Lecture Notes in Computer Science 343, 191–203, 1989.

[7] Berthold Hoffmann and Detlef Plump. Implementing term rewriting by jungle evaluation. *RAIRO Theoretical Informatics and Applications*, 25(5):445–472, 1991.

[8] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebras*, pages 263–297. Pergamon Press, 1970.

[9] Madala R.K. Krishna Rao. Modular aspects of term graph rewriting. *Theoretical Computer Science*, 208(1-2):59–86, 1998.

[10] Detlef Plump. Implementing term rewriting by graph reduction: Termination of combined systems. In *Proc. Conditional and Typed Rewriting Systems*, volume 516 of *Lecture Notes in Computer Science*, pages 307–317. Springer-Verlag, 1991.

[11] Detlef Plump. Collapsed tree rewriting: Completeness, confluence, and modularity. In *Proc. Conditional Term Rewriting Systems*, volume 656 of *Lecture Notes in Computer Science*, pages 97–112. Springer-Verlag, 1993.

[12] Detlef Plump. *Evaluation of Functional Expressions by Hypergraph Rewriting.* Doctoral dissertation, Universität Bremen, Fachbereich Mathematik und Informatik, 1993.

[13] Detlef Plump. Critical pairs in term graph rewriting. In *Proc. Mathematical Foundations of Computer Science 1994*, volume 841 of *Lecture Notes in Computer Science*, pages 556–566. Springer-Verlag, 1994.

[14] Detlef Plump. Simplification orders for term graph rewriting. In *Proc. Mathematical Foundations of Computer Science 1997*, volume 1295 of *Lecture Notes in Computer Science*, pages 458–467. Springer-Verlag, 1997.

[15] Detlef Plump. Term graph rewriting. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 2, chapter 1, pages 3–61. World Scientific, 1999.

[16] Yoshihito Toyama. Counterexamples to termination for the direct sum of term rewriting systems. *Information Processing Letters*, 25:141–143, 1987.

[17] Yoshihito Toyama. On the Church-Rosser property for the direct sum of term rewriting systems. *Journal of the ACM*, 34(1):128–143, 1987.