
Article:
Plump, Detlef, Ariola, Zena M. and Klop, Jan Willem (1997) Confluent Rewriting of Bisimilar Term Graphs. Electronic Notes in Theoretical Computer Science.

https://doi.org/10.1016/S1571-0661(05)80463-3

Electronic Notes in Theoretical Computer Science 7 (1997)
URL: http://www.elsevier.nl/locate/entcs/volume7.html 18 pages

Confluent Rewriting of Bisimilar Term Graphs [1]

Zena M. Ariola [a], Jan Willem Klop [b] and Detlef Plump [b,2]

[a] *Computer and Information Science Department, University of Oregon, Eugene, OR 97401, USA*

[b] *CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands*

Abstract

We present a survey of confluence properties of (acyclic) term graph rewriting. Results and counterexamples are given for four different kinds of term graph rewriting: besides plain applications of rewrite rules, extensions with the operations of collapsing and copying, and with both operations together are considered. Collapsing and copying together constitute bisimilarity of term graphs. We establish sufficient conditions for, and counterexamples to, confluence and confluence modulo bisimilarity of term graph rewriting over various classes of term rewriting systems.

1 Introduction

Computations with term rewrite rules play an important role in areas like functional programming, symbolic computation and theorem proving. Such computations are commonly implemented on graph-like data structures for expressions. This makes it possible to *share* common subexpressions, thereby avoiding repeated evaluations of the same subexpression.

Term graph rewriting originates from the demand for a computational model that allows to reason about implementations with sharing. In this model, rewrite rules operate on graphs rather than on trees. Although term graph rewriting is closely related to term rewriting, the two models differ with respect to important properties like termination and confluence.

In this paper, we give a survey of confluence properties of (acyclic) term graph rewriting. We address not only rewriting by applications of term rewrite rules, but also extensions with the operations of collapsing and copying, and with both operations together. These operations are important for completeness reasons: while collapsing is necessary to cope with term rewrite rules

[1] The work of J.W. Klop and D. Plump has been partially supported by the HCM Network EXPRESS.

[2] On leave from Universität Bremen, Germany.

having repeated variables in their left-hand sides, copying is needed to simulate certain term rewriting derivations that are otherwise prevented by sharing. Moreover, collapsing increases the degree of sharing and thus can considerably speed-up the evaluation process in certain cases.

When collapsing and copying are present together, the (reflexive-transitive closure of the) term graph rewrite relation contains *bisimilarity* of term graphs. We call two term graphs bisimilar if they represent the same term. Equivalently, both graphs collapse to a common term graph or yield a common term graph by copying. We investigate, in addition to confluence, under which conditions term graph rewriting is confluent modulo bisimilarity.

The rest of this paper is organized as follows. In Section 2, we introduce term graphs, collapsing, copying and bisimilarity. Section 3 contains a brief review of term graph rewriting. The relation between confluence and confluence modulo bisimilarity is clarified in Section 4. In Section 5, we recall some confluence results for non-overlapping rewrite rules and show that the full substitution strategy is cofinal. Then counterexamples are given demonstrating that the addition of collapsing or copying causes non-confluence. Orthogonal rewrite systems are treated in Section 6. It is shown that collapsing may still result in non-confluence, while plain term graph rewriting turns out to be confluent modulo bisimilarity. Section 7 is devoted to general systems with possibly overlapping rules. We present conditions under which confluence of term rewriting induces confluence or confluence modulo bisimilarity of term graph rewriting. Finally, in Section 8, we summarize our positive and negative results in two tables.

2 Term graphs and bisimilarity

Let Σ be a set of *function symbols* where each $f \in \Sigma$ comes with a natural number $\mathrm{arity}(f) \geq 0$. Function symbols of arity 0 are called *constants*. We further assume that there is an infinite set X of *variables* such that $\mathrm{X} \cap \Sigma = \emptyset$, and we set $\mathrm{arity}(x) = 0$ for each variable x.

A *hypergraph* over $\Sigma \cup \mathrm{X}$ 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 $\mathrm{lab}_G \colon \mathrm{E}_G \to \Sigma \cup \mathrm{X}$, 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 $1 + \mathrm{arity}(\mathrm{lab}_G(e))$. In the following, we call hypergraphs and hyperedges simply 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 1 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.

Instead of using hypergraphs, term graphs can alternatively be defined as directed acyclic graphs consisting of a set of labelled nodes together with a successor function from nodes to tuples of nodes (see for example [2,7]). This kind of definition is equivalent to the present one since every term graph defined in that way can be easily transformed into a hypergraph conforming to Definition 2.1, and vice versa. In this paper, we use the hypergraph framework in order to be consistent with [10,11].

A *term* over $\Sigma \cup \mathrm{X}$ 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. We write $\mathrm{term}(G)$ 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 $G \cong H$.

Definition 2.3 (Collapsing and copying) 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*. The inverse relation of collapsing is called *copying* and is denoted by $\preceq$. Proper copying, denoted by $\prec$, is the inverse relation of proper collapsing.

Two examples of collapsing and copying are given in Figure 1. It is easy to see that the collapse morphisms are the surjective graph morphisms between term graphs, and that the following fact holds.

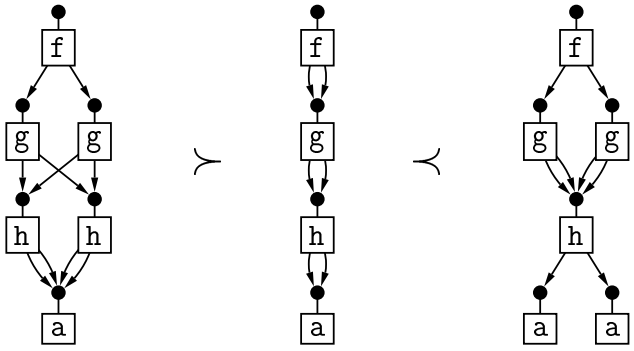

Fig. 1. Collapsing and copying

Fact 2.4 *For all term graphs G and H, $G \succeq H$ implies* $\mathrm{term}(G) = \mathrm{term}(H)$.

Definition 2.5 (Fully collapsed term graph and tree) 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 $H \succ G$.

For example, the middle graph in Figure 1 is fully collapsed. For every term graph G there are a tree $\triangle G$ and a fully collapsed term graph $\triangledown G$ such that $\mathrm{term}(G) = \mathrm{term}(\triangle G) = \mathrm{term}(\triangledown G)$, where $\triangle G$ and $\triangledown G$ are unique up to isomorphism [11].

Fact 2.6 ([11]) *For every term graph G, $\triangle G \succeq G \succeq \triangledown G$.*

Definition 2.7 (Bisimilarity) Two term graphs G and H are *bisimilar*, denoted by $G \sim H$, if $\mathrm{term}(G) = \mathrm{term}(H)$.

The three graphs in Figure 1, for instance, are bisimilar. Note that the two outer graphs are neither related by collapsing nor by copying.

Originally, the notion of bisimilarity and bisimulation was formulated in the theory of concurrent or communicating systems, also called process algebra. As it turned out, the notion applies directly and elegantly to term graphs, in order to give an equivalent formulation of "tree equivalence", that is, identity of the possibly infinite trees arising after unwinding a (possibly cyclic) term graph (see [1]). Bisimilarity and bisimulations are in the term graph setting much simpler than in process algebra, because the sum operator $(+)$ for processes is idempotent, associative, and commutative, so the objects are (edge-labelled) trees where the order of subtrees is irrelevant, other than in the case of term graphs. Our present setting of acyclic term graphs is even more simple, and enables us to define bisimilarity directly without mentioning the notion of bisimulation.

The uniqueness of trees and fully collapsed term graphs yields the following characterization of bisimilarity.

Fact 2.8 *For all term graphs G and H:*

$$G \sim H \text{ iff } \triangle G \cong \triangle H \text{ iff } \triangledown G \cong \triangledown H.$$

If we consider term graphs as "abstract graphs", i.e. as isomorphism classes of graphs, then $\succeq$ is a partial order and the equivalence class of a term graph G with respect to $\sim$ (the *bisimilarity class* of G) is in fact a complete lattice with top element $\triangle G$ and bottom element $\triangledown G$. This is shown in [1] (for possibly cyclic term graphs) in a different technical framework.

3 Term graph rewriting

We briefly review how term graphs are transformed by applications of term rewrite rules. The definition of rewrite steps given below is adopted from [10,11]. It is—as far as acyclic term graphs are concerned—equivalent to the corresponding definitions in [2,7,1].

A *term rewrite rule* $l \to r$ consists of two terms l and r over $\Sigma \cup \mathrm{X}$ 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*. We assume that the reader is familiar with basic concepts of term rewriting (see [3,8] for overviews). For the following we fix an arbitrary term rewiting system $\mathcal{R}$. The term rewrite relation associated with $\mathcal{R}$ is denoted by $\to$, its transitive closure by $\to^+$, and its reflexive-transitive closure by $\to^*$.

Given a term t, we write $\Diamond t$ for a term graph representing t such that only variables are shared, that is, each node v with an indegree greater than one satisfies $\mathrm{term}_{\Diamond t}(v) \in \mathrm{X}$, and for each variable x in t there is a unique node v with $\mathrm{term}_{\Diamond t}(v) = x$. The graph resulting from $\Diamond t$ after removing all edges labelled with variables is denoted by $\underline{\Diamond t}$. For example, Figure 2 shows the graphs $\Diamond$`f(x,x)` and $\underline{\Diamond\texttt{f(x,x)}}$. 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 3.1 (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 isomorphic to the term graph G_3 constructed as follows:

(1) $G_1 = G - \{e\}$ is the graph obtained from G by removing the unique edge e satisfying $\mathrm{res}(e) = v$.

(2) G_2 is the graph obtained from the disjoint union $G_1 + \underline{\Diamond r}$ by
- identifying v with $\mathrm{root}_{\Diamond r}$,
- identifying the image of $\mathrm{res}(e_1)$ with $\mathrm{res}(e_2)$, for each pair $\langle e_1, e_2\rangle \in \mathrm{E}_{\Diamond l} \times \mathrm{E}_{\Diamond r}$ with $\mathrm{lab}_{\Diamond l}(e_1) = \mathrm{lab}_{\Diamond r}(e_2) \in \mathrm{X}$.

(3) $G_3 = G_2[\mathrm{root}_G]$ is the term graph obtained from G_2 by removing all nodes and edges not reachable from root_G ("garbage collection").

We denote such a rewrite step also by $G \Rightarrow_v H$ or simply by $G \Rightarrow H$, and we write $G \Rightarrow^* H$ if $G \cong H$ or if there is a sequence $G = G_0 \Rightarrow G_1 \Rightarrow \ldots \Rightarrow G_n = H$.

Term graph rewriting is sound with respect to term rewriting in the fol-

Fig. 2. The graphs ◊f(x,x) and ◊f(x,x)

lowing sense.

Theorem 3.2 (Soundness [2,4]) *For all term graphs G and H,*

$$G \Rightarrow H \text{ implies } term(G) \rightarrow^+ term(H).$$

In the sequel we consider not only term graph rewriting by $\Rightarrow$ but also extensions with collapsing and copying. To this end we introduce three extensions of $\Rightarrow$.

Definition 3.3 ($\Rightarrow_{\text{coll}}$, $\Rightarrow_{\text{copy}}$, $\Rightarrow_{\text{bi}}$) The relations $\Rightarrow_{\text{coll}}$, $\Rightarrow_{\text{copy}}$ and $\Rightarrow_{\text{bi}}$ on term graphs are defined as follows:

$$\begin{aligned} \Rightarrow_{\text{coll}} &= \Rightarrow \cup \succ, \\ \Rightarrow_{\text{copy}} &= \Rightarrow \cup \prec, \\ \Rightarrow_{\text{bi}} &= \Rightarrow \cup \succ \cup \prec . \end{aligned}$$

We refer to $\Rightarrow$, $\Rightarrow_{\text{coll}}$, $\Rightarrow_{\text{copy}}$ and $\Rightarrow_{\text{bi}}$ as *plain term graph rewriting*, *term graph rewriting with collapsing*, *term graph rewriting with copying*, and *term graph rewriting with collapsing and copying*, respectively. By a *term graph rewrite relation* we mean any of these four relations.

The relations $\Rightarrow^*_{\text{coll}}$, $\Rightarrow^*_{\text{copy}}$ and $\Rightarrow^*_{\text{bi}}$ are defined analogously to $\Rightarrow^*$. Note that $\Rightarrow^*_{\text{bi}}$ contains bisimilarity since $G \sim H$ implies $G \preceq \triangle G \succeq H$ (see Fact 2.6 and 2.8). Moreover, $\Rightarrow_{\text{coll}}$, $\Rightarrow_{\text{copy}}$ and $\Rightarrow_{\text{bi}}$ are sound in the sense of Theorem 3.2 if we replace $\rightarrow^+$ by $\rightarrow^*$ (collapse and copy steps do not change the represented term).

Collapsing and copying are important because they make term graph rewriting a complete implementation of term rewriting: Theorem 7.1 will show that every term rewriting sequence can be simulated if both operations are available. On the other hand, the generality of the relation $\Rightarrow_{\text{bi}}$ causes problems with respect to termination and efficiency. Most noticeably, $\Rightarrow_{\text{bi}}$ is non-terminating for every term graph representing a term containing two or more occurrences of some subterm. This is because such a graph admits an infinite sequence of alternating collapse and copy steps. In contrast, $\Rightarrow$, $\Rightarrow_{\text{coll}}$ and $\Rightarrow_{\text{copy}}$ are terminating whenever the term rewrite relation $\rightarrow$ is terminating [3]. Another drawback is that the search space for computing a term normal

[3] A binary relation $\rightarrow$ on a set A is *terminating* (or *strongly normalizing*) if there exists no infinite sequence $a_1 \rightarrow a_2 \rightarrow \ldots$ over A.

form by $\Rightarrow_{\mathrm{bi}}$ may be much larger than for $\Rightarrow$ or $\Rightarrow_{\mathrm{coll}}$. (A detailed discussion of completeness and efficiency issues is beyond the scope of this paper. See [9,13] for conditions under which $\Rightarrow_{\mathrm{coll}}$ suffices to compute term normal forms.) For these reasons, we are interested in all four relations $\Rightarrow$, $\Rightarrow_{\mathrm{coll}}$, $\Rightarrow_{\mathrm{copy}}$ and $\Rightarrow_{\mathrm{bi}}$.

4 Confluence vs. confluence modulo bisimilarity

In this section we show by two counterexamples that for plain term graph rewriting, confluence and confluence modulo bisimilarity are incomparable properties in general. The two properties become equivalent, however, as soon as rewriting is extended by collapsing or copying.

Given term graphs G and H, and a term graph rewrite relation $\Rrightarrow$, we write $G \Rrightarrow^{\lambda} H$ if $G \Rrightarrow H$ or $G \cong H$. The inverse of a relation $\Rrightarrow$ is denoted by $\Lleftarrow$ and its symmetric closure by $\Longleftrightarrow$.

Definition 4.1 (Confluence properties) A term graph rewrite relation $\Rrightarrow$ is *subcommutative* if for all term graphs G, G_1 and G_2, $G_1 \Lleftarrow G \Rrightarrow G_2$ implies that there is a term graph G_3 such that $G_1 \Rrightarrow^{\lambda} G_3 \Lleftarrow^{\lambda} G_2$. The relation $\Rrightarrow$ is *confluent* if for every constellation $G_1 \Lleftarrow^{*} G \Rrightarrow^{*} G_2$ there is a term graph G_3 such that $G_1 \Rrightarrow^{*} G_3 \Lleftarrow^{*} G_2$. The relation $\Rrightarrow$ is *confluent modulo bisimilarity* if whenever $G_1 \Lleftarrow^{*} G \sim H \Rrightarrow^{*} H_1$, there are term graphs G_2 and H_2 such that $G_1 \Rrightarrow^{*} G_2 \sim H_2 \Lleftarrow^{*} H_1$.

It is well known that subcommutativity implies confluence (for arbitrary binary relations), see for example [8]. An important consequence of confluence is that rewriting yields deterministic results. Call a term graph N a *normal form* with respect to a relation $\Rrightarrow$ if there is no N' with $N \Rrightarrow N'$. *Uniqueness of normal forms* means that whenever $N_1 \Lleftarrow^{*} G \Rrightarrow^{*} N_2$ for normal forms N_1 and N_2, then $N_1 \cong N_2$. While confluence implies uniqueness of normal forms, confluence modulo bisimilarity implies uniqueness of normal forms up to bisimilarity.

For plain term graph rewriting, confluence need not imply confluence modulo bisimilarity, and vice versa. This is shown by the next two examples.

Example 4.2 Consider the following system [4]:

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

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

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

It is easy to check that $\Rightarrow$ is confluent, but Figure 3 shows that confluence modulo bisimilarity fails.

[4] This system is used in [7] for a different purpose.

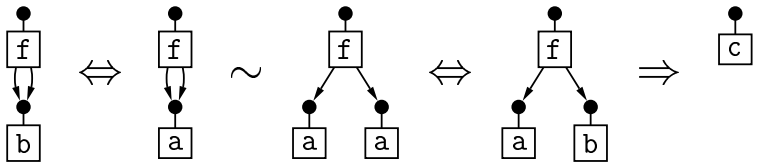

Fig. 3. Confluence without confluence modulo bisimilarity

Example 4.3 For $\Rightarrow$, confluence modulo bisimilarity does not imply confluence in general. This is demonstrated by the following system:

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

Theorem 7.6 will show that $\Rightarrow$ is confluent modulo bisimilarity, since $\Rightarrow$ is weakly normalizing and term rewriting is confluent. However, the term graph representing $\mathtt{g(a)}$ has two non-isomorphic normal forms (see Figure 4), hence $\Rightarrow$ is not confluent.

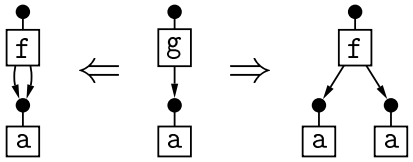

Fig. 4. Confluence modulo bisimilarity without confluence

We now show that in the presence of collapsing or copying, confluence and confluence modulo bisimilarity become equivalent.

Theorem 4.4 *Let* $\Rrightarrow \in \{\Rightarrow_{\mathrm{coll}}, \Rightarrow_{\mathrm{copy}}, \Rightarrow_{\mathrm{bi}}\}$. *Then* $\Rrightarrow$ *is confluent if and only if* $\Rrightarrow$ *is confluent modulo bisimilarity.*

Proof. First we consider the case of $\Rightarrow_{\mathrm{coll}}$. "If": Let $G_1 \Leftarrow^*_{\mathrm{coll}} G \Rightarrow^*_{\mathrm{coll}} G_2$ be two term graph derivations. Confluence modulo bisimilarity implies that there are term graphs G_3 and G_4 such that $G_1 \Rightarrow^*_{\mathrm{coll}} G_3 \sim G_4 \Leftarrow^*_{\mathrm{coll}} G_2$. By Fact 2.6 and 2.8, $G_3 \succeq \nabla G_3 \preceq G_4$. Hence $G_1 \Rightarrow^*_{\mathrm{coll}} \nabla G_3 \Leftarrow^*_{\mathrm{coll}} G_2$. "Only if": Suppose that $G_1 \Leftarrow^*_{\mathrm{coll}} G \sim H \Rightarrow^*_{\mathrm{coll}} H_1$. Then $G \preceq \triangle G \succeq H$ and hence $G_1 \Leftarrow^*_{\mathrm{coll}} \triangle G \Rightarrow^*_{\mathrm{coll}} H_1$. By confluence, there is a term graph G_2 such that $G_1 \Rightarrow^*_{\mathrm{coll}} G_2 \Leftarrow^*_{\mathrm{coll}} H_1$. Thus, $\Rightarrow_{\mathrm{coll}}$ is confluent modulo bisimilarity.

When considering $\Rightarrow_{\mathrm{copy}}$ instead of $\Rightarrow_{\mathrm{coll}}$, we just exchange $\succeq$ and $\preceq$ in the above proof, replace ∇G_3 by $\triangle G_3$, and $\triangle G$ by ∇G.

Finally, it is obvious that the proof also works for $\Rightarrow_{\mathrm{bi}}$. □

5 Non-overlapping systems

It is known that plain term graph rewriting is confluent—in fact subcommutative—if the left-hand sides of the given term rewrite rules do not overlap. After recalling this and a related result about the uniqueness of complete developments, we show that the reduction strategy of full substitution is cofinal. Then counterexamples are given demonstrating that confluence fails as soon as term graph rewriting is extended with copying or collapsing.

Definition 5.1 (Non-overlapping) A term s *overlaps* a term t in a subterm u of t if u is not a variable and if there are substitutions [5] σ and τ such that $\sigma(s) = \tau(u)$. The term rewriting system $\mathcal{R}$ is *non-overlapping* if for all rules $l_1 \to r_1$ and $l_2 \to r_2$ in $\mathcal{R}$, l_1 overlaps l_2 in a subterm u only if $u = l_2$ and $(l_1 \to r_1) = (l_2 \to r_2)$.

Theorem 5.2 *If $\mathcal{R}$ is non-overlapping, then $\Rightarrow$ is subcommutative.*

Proof. A proof is already given in [14], in a slightly different technical framework. A proof conforming to the present setting can be found in [11], as part of the proof of the so-called Critical Pair Lemma. □

For the rest of this section we assume that $\mathcal{R}$ is an arbitrary non-overlapping system. The following property of subcommutative relations will be needed in showing that the full substitution strategy is cofinal.

Corollary 5.3 *For all term graphs G, G_1 and G_2, $G_1 \Leftarrow G \Rightarrow^* G_2$ implies that there is a term graph G_3 such that $G_1 \Rightarrow^* G_3 \Leftarrow^\lambda G_2$.*

Proof. By induction on the length of $G \Rightarrow^* G_2$, using subcommutativity. □

We are going to show that complete developments of sets of redexes yield unique results. This fact allows to define the full substitution strategy. In the next section, the cofinality property of this strategy will be used to prove that $\Rightarrow$ is confluent modulo bisimilarity over orthogonal rewrite systems.

Definition 5.4 (Redex) A node v in a term graph G is a *redex* if there is a rule $l \to r$ in $\mathcal{R}$ such that $G[v]$ is an instance of $\Diamond l$.

Hence, v is a redex if and only if $G \Rightarrow_{v, l\to r} H$ for some rule $l \to r$ and term graph H. Since $\mathcal{R}$ is non-overlapping, node v uniquely determines the rule $l \to r$.

Given a term graph rewrite step $G \Rightarrow H$ and a node v in G, v either has a unique image in H or is removed by garbage collection. We use a partial function $\mathrm{tr}_{G\Rightarrow H}\colon \mathrm{V}_G \to \mathrm{V}_H$, the *track function* associated with $G \Rightarrow H$, to assign to each node of G its corresponding node in H. The track function associated with a sequence of rewrite steps is obtained in an obvious way by composition. A formal definition of the track function can be found in [11].

[5] A *substitution* σ is a mapping on the set of terms over $\Sigma \cup \mathrm{X}$ such that $\sigma(c) = c$ for every constant c, and $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$ for every composite term $f(t_1, \ldots, t_n)$.

Definition 5.5 (Residuals) Let Π be a set of redexes in a term graph G. The set $\rho(\Pi)$ of *residuals* of Π with respect to a rewrite sequence $G \Rightarrow^* H$ is defined as follows. If $G \Rightarrow^* H$ has length 0, then $\rho(\Pi) = i(\Pi)$ for the unique isomorphism $i: G \to H$. If $G \Rightarrow^* H$ has the form $G \Rightarrow_v G' \Rightarrow^* H$, then $\rho(\Pi)$ is the set of residuals of $\mathrm{tr}_{G \Rightarrow G'}(\Pi - \{v\})$ with respect to $G' \Rightarrow^* H$.

By the assumption that $\mathcal{R}$ is non-overlapping, the residuals of a redex set are again redexes. Note that this is different in term rewriting: there this property may fail when rules are present that have repeated variables in their left-hand sides.

Definition 5.6 (Development) A *development* of a set Π of redexes in a term graph G is either a derivation $G \Rightarrow^* H$ of length 0, or a derivation of the form $G \Rightarrow_{v, l \to r} G' \Rightarrow^* H$ such that $v \in \Pi$ and such that $G' \Rightarrow^* H$ is a development of the residuals of Π in G'. The development is *complete* if Π has no residuals in H.

Theorem 5.7 (Uniqueness of developments) [6] *Given a set Π of redexes in a term graph G, all complete developments of Π end (up to isomorphism) in the same term graph.*

Proof. Consider two complete developments $G \Rightarrow^* H_1$ and $G \Rightarrow^* H_2$ of Π. We proceed by induction on the number of redexes in Π. If Π is empty, then $H_1 \cong G \cong H_2$ by the definition of complete development. Otherwise, there are (not necessarily distinct) nodes v_1 and v_2 in Π such that for i=1,2, $G \Rightarrow^* H_i$ has the form $G \Rightarrow_{v_i} G_i \Rightarrow^* H_i$. By the proof of Theorem 5.2, there are steps $G_1 \Rightarrow^{\lambda}_{w_2} G' \Leftarrow^{\lambda}_{w_1} G_2$ such that w_1 and w_2 are residuals of v_1 and v_2, respectively, and such that the residuals of Π with respect to $G \Rightarrow_{v_1} G_1 \Rightarrow^{\lambda}_{w_2} G'$ and $G \Rightarrow_{v_2} G_2 \Rightarrow^{\lambda}_{w_1} G'$ are the same (see [11]). Now let $G' \Rightarrow^* H$ be a complete development of the redex set $\Pi' = \mathrm{tr}_{G \Rightarrow G_1 \Rightarrow^{\lambda} G'}(\Pi - \{v_1, v_2\}) = \mathrm{tr}_{G \Rightarrow G_2 \Rightarrow^{\lambda} G'}(\Pi - \{v_1, v_2\})$. Then both $G_1 \Rightarrow^* H_1$ and $G_1 \Rightarrow^{\lambda}_{w_2} G' \Rightarrow^* H$ are complete developments of $\mathrm{tr}_{G \Rightarrow G_1}(\Pi - \{v_1\})$. Hence, by induction hypothesis, $H_1 \cong H$. Analogously one shows $H_2 \cong H$. Thus $H_1 \cong H_2$. □

Given a term graph G, we denote by $\mathrm{Cpl}(G)$ a term graph that results from a complete development of all redexes in G. The process of repeatedly developing all redexes is called the *full substitution* or *Gross-Knuth* strategy in the context of term rewriting systems (see [8]). We show that this strategy is "cofinal" for term graph rewriting over non-overlapping systems.

Theorem 5.8 (Cofinality) *For all term graphs G and H, $G \Rightarrow^* H$ implies that there is $n \geq 0$ such that $H \Rightarrow^* \mathrm{Cpl}^n(G)$.*

Proof. By induction on the length of $G \Rightarrow^* H$. Suppose that $G \Rightarrow^* H' \Rightarrow H$ for some term graph H'. By induction hypothesis, $H' \Rightarrow^* \mathrm{Cpl}^n(G)$ for some $n \geq 0$. Then, by Corollary 5.3, there is a term graph H'' such that

[6] This result appears in [2] without proof.

$H \Rightarrow^* H'' \Leftarrow^\lambda \mathrm{Cpl}^n(G)$. Thus, by the definition of complete development, $H'' \Rightarrow^* \mathrm{Cpl}^{n+1}(G)$. It follows $H \Rightarrow^* \mathrm{Cpl}^{n+1}(G)$. □

Confluence of $\Rightarrow$ does no longer hold if collapsing or copying is added, as the following two counterexamples demonstrate. Moreover, the examples show that none of the four relations $\Rightarrow$, $\Rightarrow_{\mathrm{coll}}$, $\Rightarrow_{\mathrm{copy}}$ and $\Rightarrow_{\mathrm{bi}}$ is confluent modulo bisimilarity for non-overlapping systems in general.

Example 5.9 Consider the following non-overlapping system of Huet [5]:

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

Figure 5 demonstrates that the tree representing $\mathtt{f(c,c)}$ has two non-isomorphic normal forms with respect to $\Rightarrow_{\mathrm{coll}}$, so $\Rightarrow_{\mathrm{coll}}$ and $\Rightarrow_{\mathrm{bi}}$ are neither confluent nor confluent modulo bisimilarity. Note that the left-hand sides of the first two rewrite rules contain the variable `x` twice.

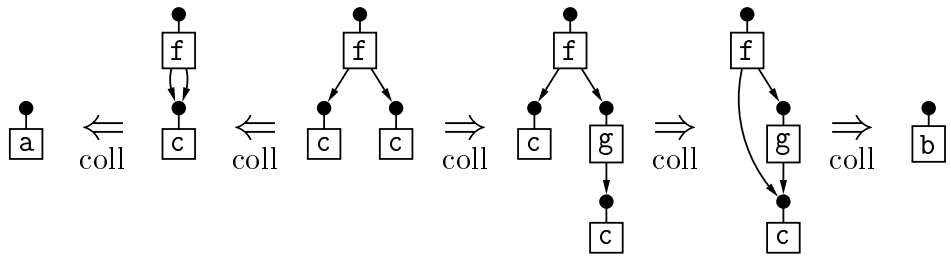

Fig. 5. Non-confluence of $\Rightarrow_{\mathrm{coll}}$ and $\Rightarrow_{\mathrm{bi}}$

Example 5.10 The rule

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

also contains two occurrences of `x` in its left-hand side. It shows that, for non-overlapping systems, $\Rightarrow_{\mathrm{copy}}$ need neither be confluent nor confluent modulo bisimilarity. To see this, observe that the graph on the right in Figure 6 is a normal form with respect to $\Rightarrow_{\mathrm{copy}}$.

Figure 6 also demonstrates that plain term graph rewriting is not confluent modulo bisimilarity for non-overlapping systems in general. This is because the rewrite step on the left is proper, and the graphs in the middle and on the right are bisimilar.

6 Orthogonal systems

The counterexamples of the previous section show that for non-overlapping systems, $\Rightarrow$, $\Rightarrow_{\mathrm{coll}}$, $\Rightarrow_{\mathrm{copy}}$ and $\Rightarrow_{\mathrm{bi}}$ need not be confluent modulo bisimilarity,

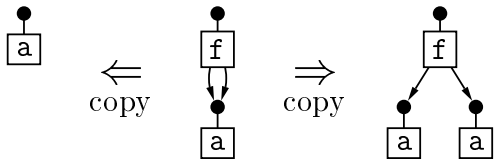

Fig. 6. Non-confluence of $\Rightarrow_{\mathrm{copy}}$

and the last three relations need neither be confluent. In this section and the next it will become clear that this failure is caused, with the exception of $\Rightarrow_{\mathrm{coll}}$, by rewrite rules with repeated variables in their left-hand sides.

Definition 6.1 (Orthogonal) The term rewriting system $\mathcal{R}$ is *left-linear* if for each rewrite rule $l \to r$ in $\mathcal{R}$, no variable occurs more than once in l. The system $\mathcal{R}$ is *orthogonal* if it is left-linear and non-overlapping.

The main result of this section is that for orthogonal systems, plain term graph rewriting is confluent modulo bisimilarity. As far as confluence is concerned, we know from the previous section that $\Rightarrow$ is confluent for non-overlapping systems and hence, in particular, for orthogonal systems. In the next section it is shown that $\Rightarrow_{\mathrm{copy}}$ and $\Rightarrow_{\mathrm{bi}}$ are confluent for classes of systems that properly include all orthogonal systems. In contrast, term graph rewriting with collapsing need not be confluent even for orthogonal systems.

Example 6.2 Consider the single rule

$$\mathtt{c} \to \mathtt{g}(\mathtt{c})$$

and suppose that Σ contains a binary function symbol $\mathtt{f}$. Figure 7 shows two $\Rightarrow_{\mathrm{coll}}$-derivations starting from the tree representing $\mathtt{f}(\mathtt{c},\mathtt{c})$ such that the resulting graphs do not have a common reduct under $\Rightarrow_{\mathrm{coll}}$: the graphs derivable on the left represent the terms $\mathtt{f}(\mathtt{g}^n(\mathtt{c}),\mathtt{g}^n(\mathtt{c}))$, $n \geq 1$, while the graphs derivable on the right represent $\mathtt{f}(\mathtt{g}^n(\mathtt{c}),\mathtt{g}^{n+1}(\mathtt{c}))$, $n \geq 0$. Thus $\Rightarrow_{\mathrm{coll}}$ is non-confluent.

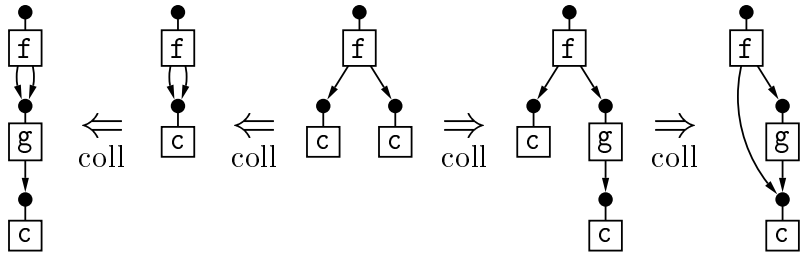

Fig. 7. Non-confluence of $\Rightarrow_{\mathrm{coll}}$

Lemma 6.3 *Let $\mathcal{R}$ be orthogonal. Then for all term graphs G and H, $G \sim H$ implies* $\mathrm{Cpl}(G) \sim \mathrm{Cpl}(H)$.

Proof. Given a complete development $T \Rightarrow^* \mathrm{Cpl}(T)$ of all redexes in a term graph T, there is a corresponding complete development $\mathrm{term}(T) \rightarrow^* \mathrm{term}(\mathrm{Cpl}(T))$ of all redexes in $\mathrm{term}(T)$ (see [7] for a proof in a slightly different technical setting). Since for orthogonal term rewriting systems, all complete developments of a set of redexes yield the same result [6], $\mathrm{term}(G) = \mathrm{term}(H)$ implies $\mathrm{term}(\mathrm{Cpl}(G)) = \mathrm{term}(\mathrm{Cpl}(H))$. □

It is worth mentioning that this lemma does not hold for non-overlapping systems. A counterexample is again $\mathcal{R} = \{\mathtt{f}(\mathtt{x},\mathtt{x}) \rightarrow \mathtt{a}\}$: in Figure 6, the graph in the middle is bisimilar to the graph on the right which is a normal form with respect to $\Rightarrow$.

Theorem 6.4 *If $\mathcal{R}$ is orthogonal, then $\Rightarrow$ is confluent modulo bisimilarity.*

Proof. Suppose that $G_1 \Leftarrow^* G \sim H \Rightarrow^* H_1$. By Theorem 5.8, there are $m, n \geq 0$ such that $G_1 \Rightarrow^* \mathrm{Cpl}^m(G)$ and $H_1 \Rightarrow^* \mathrm{Cpl}^n(H)$. Hence, choosing $p = max(m, n)$, we obtain $G_1 \Rightarrow^* \mathrm{Cpl}^p(G)$ and $H_1 \Rightarrow^* \mathrm{Cpl}^p(H)$. Now $\mathrm{Cpl}^p(G) \sim \mathrm{Cpl}^p(H)$ follows from Lemma 6.3. □

7 General systems

In this section we drop the assumption of the two previous sections that $\mathcal{R}$ is non-overlapping. Instead, we infer confluence of $\Rightarrow_{\mathrm{bi}}$, $\Rightarrow_{\mathrm{copy}}$ and $\Rightarrow_{\mathrm{coll}}$ from confluence of term rewriting. In the case of $\Rightarrow_{\mathrm{copy}}$ and $\Rightarrow_{\mathrm{coll}}$, this requires suitable further conditions. Finally, we give sufficient conditions under which confluence of term rewriting makes $\Rightarrow$ confluent modulo bisimilarity.

In the sequel we write $\triangle t$ for the tree representation of a term t. (Hence $\triangle G \cong \triangle \mathrm{term}(G)$ for every term graph G.)

We first show that term graph rewriting with collapsing and copying can simulate term rewriting, following the proof of the so-called Completeness Theorem in [11].

Theorem 7.1 *For all term graphs G and H:*

$$G \Rightarrow_{\mathrm{bi}}^* H \text{ if and only if } term(G) \rightarrow^* term(H).$$

Proof. "Only if": By soundness of $\Rightarrow_{\mathrm{bi}}$.
"If": Suppose that for every term rewrite step $t \rightarrow u$ there are term graphs T and U such that

$$\triangle t \succeq T \Rightarrow U \preceq \triangle u. \tag{1}$$

Then $term(G) \rightarrow^* term(H)$ implies $\triangle term(G) \Rightarrow_{\mathrm{bi}}^* \triangle term(H)$, and with Fact 2.6 follows $G \preceq \triangle term(G) \Rightarrow_{\mathrm{bi}}^* \triangle term(H) \succeq H$. To show (1), let $l \rightarrow r$ be the rule applied in $t \rightarrow u$ and π be the associated redex position in t. Let v be the unique node in $\triangle t$ specified by π. Then there is a collapsing $\triangle t \succeq T$ such that $T[v']$ is fully collapsed, where v' is the image of v in T, and such that each node of T not belonging to $T[v']$ has an indegree of at most one. By the structure of T, there is a step $T \Rightarrow_{v', l \rightarrow r} U$ such that $\mathrm{term}(U) = u$. (Since $T[v']$

is fully collapsed, $l \to r$ is applicable at v' even if l contains repeated variables, and as there is a unique path from root_T to v', $T \Rightarrow_{v', l \to r} U$ simulates $t \to u$.) Hence $U \preceq \triangle u$. □

Corollary 7.2 *The relation $\Rightarrow_{\mathrm{bi}}$ is confluent if and only if $\to$ is confluent.*

Proof. "Only if": Suppose that $t_1 \leftarrow^* t \to^* t_2$ for some terms t, t_1 and t_2. Then $\triangle t_1 \Leftarrow^*_{\mathrm{bi}} \triangle t \Rightarrow^*_{\mathrm{bi}} \triangle t_2$ by Theorem 7.1. Since $\Rightarrow_{\mathrm{bi}}$ is confluent, there is a term graph G such that $\triangle t_1 \Rightarrow^*_{\mathrm{bi}} G \Leftarrow^*_{\mathrm{bi}} \triangle t_2$. Hence $t_1 \to^* \mathrm{term}(G) \leftarrow^* t_2$ by Theorem 7.1.
"If": Given derivations $G_1 \Leftarrow^*_{\mathrm{bi}} G \Rightarrow^*_{\mathrm{bi}} G_2$, Theorem 7.1 yields $\mathrm{term}(G_1) \leftarrow^* \mathrm{term}(G) \to^* \mathrm{term}(G_2)$. Then, since $\to$ is confluent, there is a term t such that $\mathrm{term}(G_1) \to^* t \leftarrow^* \mathrm{term}(G_2)$. With Theorem 7.1 follows $G_1 \Rightarrow^*_{\mathrm{bi}} \triangle t \Leftarrow^*_{\mathrm{bi}} G_2$, since $\mathrm{term}(\triangle t) = t$. □

In order to simulate term rewriting by $\Rightarrow_{\mathrm{copy}}$, the underlying system $\mathcal{R}$ has to be left-linear.

Theorem 7.3 *If $\mathcal{R}$ is left-linear, then for all terms t and u:*

$$\triangle t \Rightarrow^*_{\mathrm{copy}} \triangle u \text{ if and only if } t \to^* u.$$

Proof. "Only if": Immediate consequence of soundness of $\Rightarrow_{\mathrm{copy}}$.
"If": It suffices to show that for every term rewrite step $t \to u$ there is a term graph U such that

$$\triangle t \Rightarrow U \preceq \triangle u.$$

Let $l \to r$ be the rule applied in $t \to u$. Then, since $\mathcal{R}$ is left-linear, $\triangle t \Rightarrow_{v, l \to r} U$ for some term graph U, where v is the node corresponding to the redex position in t. As there is no sharing in $\triangle t$, we have $\mathrm{term}(U) = u$. Thus $U \preceq \triangle u$. □

Corollary 7.4 *If $\mathcal{R}$ is left-linear, then $\Rightarrow_{\mathrm{copy}}$ is confluent if and only if $\to$ is confluent.*

Proof. "Only if": Easy consequence of Theorem 7.3 and soundness of $\Rightarrow_{\mathrm{copy}}$.
"If": Consider derivations $G_1 \Leftarrow^*_{\mathrm{copy}} G \Rightarrow^*_{\mathrm{copy}} G_2$. Then, by soundness of $\Rightarrow_{\mathrm{copy}}$, $\mathrm{term}(G_1) \leftarrow^* \mathrm{term}(G) \to^* \mathrm{term}(G_2)$. Confluence of $\to$ implies that there is a term t such that $\mathrm{term}(G_1) \to^* t \leftarrow^* \mathrm{term}(G_2)$. With Theorem 7.3 follows $\triangle\mathrm{term}(G_1) \Rightarrow^*_{\mathrm{copy}} \triangle t \Leftarrow^*_{\mathrm{copy}} \triangle\mathrm{term}(G_2)$. Hence, using Fact 2.6,

$$G_1 \preceq \triangle\mathrm{term}(G_1) \Rightarrow^*_{\mathrm{copy}} \triangle t \Leftarrow^*_{\mathrm{copy}} \triangle\mathrm{term}(G_2) \succeq G_2.$$

□

Corollary 7.4 implies that $\Rightarrow_{\mathrm{copy}}$ is confluent, in particular, for orthogonal systems. For it is well known that orthogonality implies confluence of term rewriting (see for example [3,8]).

An analogue to Corollary 7.4 for the case of $\Rightarrow_{\mathrm{coll}}$ can be obtained by replacing the condition of left-linearity with weak normalization of $\Rightarrow_{\mathrm{coll}}$. A

term graph rewrite relation $\Rrightarrow$ is *weakly normalizing* if for every term graph G there is a normal form N such that $G \Rrightarrow^* N$.

Theorem 7.5 ([10]) *If $\Rightarrow_{\mathrm{coll}}$ is weakly normalizing, then $\Rightarrow_{\mathrm{coll}}$ is confluent if and only if $\rightarrow$ is confluent.*

We conclude this section by giving conditions under which confluence of $\rightarrow$ guarantees that $\Rightarrow$ is confluent modulo bisimilarity. It turns out that both left-linearity of $\mathcal{R}$ and weak normalization of $\Rightarrow$ are needed.

Theorem 7.6 *If $\mathcal{R}$ is left-linear, $\rightarrow$ confluent, and $\Rightarrow$ weakly normalizing, then $\Rightarrow$ is confluent modulo bisimilarity.*

Proof. Given a constellation $G_1 \Leftarrow^* G \sim H \Rightarrow^* H_1$, consider normal forms G_2 and H_2 of G_1 and H_1, respectively. Then $\mathrm{term}(G_2) \leftarrow^* \mathrm{term}(G) = \mathrm{term}(H) \rightarrow^* \mathrm{term}(H_2)$ by soundness of $\Rightarrow$. Since G_2 and H_2 are normal forms and $\mathcal{R}$ is left-linear, $\mathrm{term}(G_2)$ and $\mathrm{term}(H_2)$ are normal forms with respect to $\rightarrow$. (Left-linearity implies that $\Diamond l$ is a tree for every term rewrite rule $l \rightarrow r$; hence, given a term graph T, there is a graph morphism $\underline{\Diamond l} \rightarrow T$ if and only if $l \rightarrow r$ is applicable to $\mathrm{term}(T)$.) Now confluence of $\rightarrow$ yields $\mathrm{term}(G_2) = \mathrm{term}(H_2)$, thus $G_2 \sim H_2$. □

The premise of Theorem 7.6 cannot be relaxed by dropping left-linearity or weak normalization, as is witnessed by Example 5.10 and 4.2, respectively. Moreover, weak normalization of $\Rightarrow$ cannot be replaced by weak normalization of $\rightarrow$. We demonstrate this by a counterexample from [10].

Example 7.7 Suppose that $\mathcal{R}$ consists of the following rules:

$$\begin{aligned} \mathtt{f}(\mathtt{x}) &\rightarrow \mathtt{g}(\mathtt{x},\mathtt{x}) \\ \mathtt{a} &\rightarrow \mathtt{b} \\ \mathtt{g}(\mathtt{a},\mathtt{b}) &\rightarrow \mathtt{c} \\ \mathtt{g}(\mathtt{b},\mathtt{b}) &\rightarrow \mathtt{f}(\mathtt{a}) \end{aligned}$$

Using structural induction on terms, it is easy to verify that every term has a unique normal form. Hence $\rightarrow$ is weakly normalizing and confluent. But Figure 8 shows that $\Rightarrow$ is neither confluent nor confluent modulo bisimilarity. (Notice that there is no graph rewrite step $\triangledown\mathtt{g}(\mathtt{a},\mathtt{a}) \Rightarrow \triangle\mathtt{g}(\mathtt{a},\mathtt{b})$ corresponding to the term rewrite step $\mathtt{g}(\mathtt{a},\mathtt{a}) \rightarrow \mathtt{g}(\mathtt{a},\mathtt{b})$.)

We finally remark that the assumptions of Theorem 7.6 do not guarantee that $\Rightarrow$ is confluent. This can be seen from Example 4.3. There, $\Rightarrow$ is even terminating and $\mathcal{R}$ is *almost orthogonal*, that is, every two overlapping term rewrite steps $t_1 \leftarrow t \rightarrow t_2$ satisfy $t_1 = t_2$ and the overlap occurs at the roots of the left-hand sides of the applied rules.

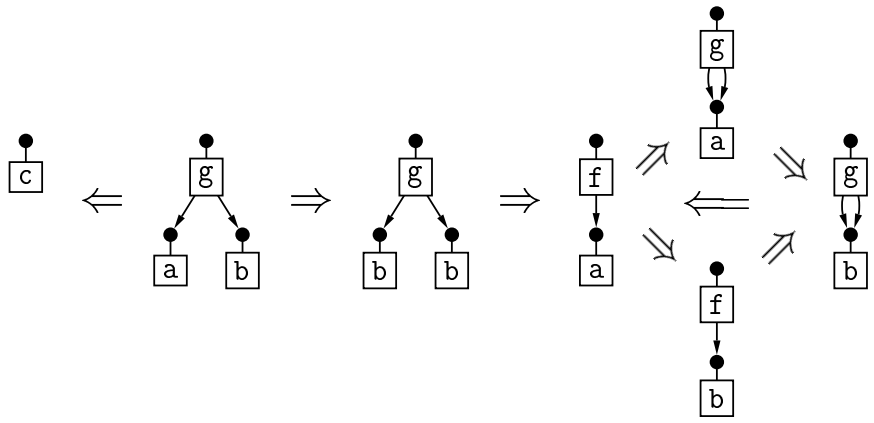

Fig. 8.

8 Conclusion

Our positive and negative results on confluence and confluence modulo bisimilarity are summarized in two tables below the references. In both tables, a "+" means that confluence resp. confluence modulo bisimilarity holds under the given conditions, while a "−" indicates that there exists a counterexample.

The conditions for confluence considered in this paper either forbid overlaps between term rewrite rules or require confluence of the associated term rewrite relation. Another tool for analyzing confluence are *critical pairs* of term graph rewrite steps. We refer to [12] for their definition and their use to decide confluence of $\Rightarrow_{\mathrm{coll}}$ in the presence of termination.

References

[1] Zena M. Ariola and Jan Willem Klop. Equational term graph rewriting. *Fundamenta Informaticae*, 26:207–240, 1996.

[2] Hendrik 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.

[3] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science,* volume B, chapter 6. Elsevier, 1990.

[4] 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.

[5] Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, 1980.

[6] Gérard Huet and Jean-Jacques Lévy. Computations in orthogonal rewrite systems, I and II. In Jean-Louis Lassez and Gordon Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*, pages 395–443. MIT Press, 1991.

[7] Richard Kennaway, Jan Willem Klop, Ronan Sleep, and Fer-Jan de Vries. On the adequacy of term graph rewriting for simulating term rewriting. *ACM Transactions on Programming Languages and Systems*, 16(3):493–523, 1994.

[8] Jan Willem Klop. Term rewriting systems. In S. Abramsky, Dov M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science,* volume 2, pages 1–116. Oxford University Press, 1992.

[9] Detlef Plump. Graph-reducible term rewriting systems. In *Proc. Graph Grammars and Their Application to Computer Science*, volume 532 of *Lecture Notes in Computer Science*, pages 622–636. Springer-Verlag, 1991.

[10] 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.

[11] Detlef Plump. Evaluation of functional expressions by hypergraph rewriting. Dissertation, Universität Bremen, Fachbereich Mathematik und Informatik, 1993.

[12] 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.

[13] Madala R.K. Krishna Rao. Graph reducibility of term rewriting systems. In *Proc. Mathematical Foundations of Computer Science 1995*, volume 969 of *Lecture Notes in Computer Science*, pages 371–381. Springer-Verlag, 1995.

[14] John Staples. Computation on graph-like expressions. *Theoretical Computer Science*, 10:171–185, 1980.

Table 1
Overview of confluence

	$\Rightarrow$	$\Rightarrow_{\text{coll}}$	$\Rightarrow_{\text{copy}}$	$\Rightarrow_{\text{bi}}$
$\mathcal{R}$ orthogonal	+	−	+	+
$\mathcal{R}$ non-overlapping	+	−	−	−
$\rightarrow$ confluent	−	−	−	+
$\mathcal{R}$ left-linear, $\rightarrow$ confluent	−	−	+	+
$\rightarrow$ confluent, $\Rightarrow_{\text{coll}}$ weakly normalizing	−	+	−	+
$\mathcal{R}$ left-linear, $\rightarrow$ confluent, $\Rightarrow$ weakly normalizing	−	+	+	+

Table 2
Overview of confluence modulo bisimilarity

	$\Rightarrow$	$\Rightarrow_{\text{coll}}$	$\Rightarrow_{\text{copy}}$	$\Rightarrow_{\text{bi}}$
$\mathcal{R}$ orthogonal	+	see Table 1		
$\mathcal{R}$ non-overlapping	−			
$\rightarrow$ confluent	−			
$\mathcal{R}$ left-linear, $\rightarrow$ confluent	−			
$\rightarrow$ confluent, $\Rightarrow_{\text{coll}}$ weakly normalizing	−			
$\mathcal{R}$ left-linear, $\rightarrow$ confluent, $\Rightarrow$ weakly normalizing	+			