
Proceedings Paper:
Plump, Detlef orcid.org/0000-0002-1148-822X (1994) Critical Pairs in Term Graph Rewriting. In: Proc. Mathematical Foundations of Computer Science (MFCS 1994). Lecture Notes in Computer Science . Springer , pp. 556-566.

https://doi.org/10.1007/3-540-58338-6_102

Critical Pairs in Term Graph Rewriting

Detlef Plump*
Universität Bremen

Abstract

Term graphs represent functional expressions such that common subexpressions can be shared, making expression evaluation more efficient than with strings or trees. Rewriting of term graphs proceeds by both applications of term rewrite rules and folding steps which enhance the degree of sharing. The present paper introduces *critical pairs* in term graph rewriting and establishes a Critical Pair Lemma as an analogue to the well-known result in term rewriting. This leads to a decision procedure for confluence in the presence of termination. As a by-product, the procedure can be used as a confluence test for term rewriting and as such it extends the classical test of Knuth and Bendix because it applies to all terminating and to certain non-terminating term rewriting systems.

1 Introduction

The rich theory of term rewriting systems is an essential tool for many developments in areas like algebraic specification, automated theorem proving, and functional programming. Implementations in these fields, however, usually relinquish pure term rewriting in form of string or tree rewriting for efficiency reasons. Instead, terms are represented by pointer structures—i.e. *graphs*—which allow to share common subterms (see e.g. [5, 13, 18]). But this changes the computational model, having consequences that may be overlooked at a first glance. For instance, the two models behave differently with respect to termination, confluence, and the combination of both properties.

The term "term graph rewriting" was introduced by Barendregt et al. [1] and is now used generically for various approaches to expression evaluation by graph rewriting such as [2, 4, 6, 7]. (See also [18] for a collection of recent papers on term graph rewriting).

This paper is concerned with the confluence property of term graph rewriting. The objective is to give a characterization and a decision procedure for confluence, in analogy to the well-known result of Knuth and Bendix for term rewriting [11]. This is achieved by introducing *critical pairs* in term graph rewriting and

*Author's address: Fachbereich Mathematik und Informatik, Universität Bremen, Postfach 33 04 40, 28334 Bremen, Germany. E-mail: det@informatik.uni-bremen.de. Research partially supported by ESPRIT Basic Research Working Group 6112, *COMPASS*.

by establishing a sufficient condition for local confluence in form of a Critical Pair Lemma. Moreover, the confluence test can also be used for term rewriting and as such extends the classical test of Knuth and Bendix since it applies not only to terminating systems but also to those non-terminating systems that become terminating under graph rewriting. Finally it is shown that term graph rewriting allows to decide equivalence of terms for a larger class of systems than in the case of term rewriting.

A distinctive feature of the present approach to term graph rewriting is the incorporation of *folding steps* which identify common subexpressions. Folding allows to handle non-left-linear rewrite systems and is necessary to make term graph rewriting a complete method for equational deduction. Moreover, it causes a vast speed-up of the evaluation process in certain examples.

In this paper, proofs are omitted for lack of space; they can be found in [16].

2 Term Graph Rewriting

Let Σ be a *signature*, i.e. a set of function symbols, and X be a set of *variables* disjoint from Σ. Each function symbol f comes with an integer $arity(f) \geq 0$; for each variable x, let $arity(x) = 0$.

A *hypergraph* G is a system $\langle V_G, E_G, s_G, t_G, l_G \rangle$, where V_G and E_G are finite sets of *nodes* (or *vertices*) and *(hyper-)edges*, $s_G: E_G \to V_G$ and $t_G: E_G \to V_G^*$ are mappings that assign a *source node* and a string of *target nodes* to each edge, and $l_G: E_G \to \Sigma \cup X$ is a mapping that labels each edge e such that $arity(l_G(e))$ is the length of $t_G(e)$.

Given two nodes v and v', write $v >_G^1 v'$ if there is an edge e with source node v and v' occuring in $t_G(e)$. The transitive (reflexive-transitive) closure of $>_G^1$ is denoted by $>_G$ ($\geq_G$). G is *acyclic* if $>_G$ is irreflexive.

A hypergraph G is a *term graph* if (1) there is a node $root_G$ such that $root_G \geq_G v$ for each node v, (2) G is acyclic, and (3) each node has a unique outgoing edge.

Figure 1 shows three term graphs with function symbols $+$, $\times$, s, and 0, where $arity(+) = arity(\times) = 2$, $arity(\mathrm{s}) = 1$, and $arity(0) = 0$. Edges are depicted as boxes with inscribed labels, and circles represent nodes. A line connects each edge with its source node, while arrows point to target nodes. The order in a target string is given by the left-to-right order of the arrows leaving a box.

Terms over Σ and X are defined as usual (see e.g. [3]). A node v in a term graph G represents the term $term_G(v) = l_G(e)(term_G(v_1), \ldots, term_G(v_n))$, where e is the unique edge with source v, and $t_G(e) = v_1 \ldots v_n$. In the following $term(G)$ stands for $term_G(root_G)$. As an example, if G is the left term graph in Figure 1, then $term(G) = +(\mathrm{s}(0), \times(\mathrm{s}(0), +(0, 0)))$.

A *rewrite rule* $l \to r$ consists of two terms l and r such that l is not a variable and all variables in r occur also in l. A set $\mathcal{R}$ of such rules is a *term rewriting system*. The reader is assumed to be familiar with basic concepts of term rewriting (see e.g. [3, 10, 14]). The rewrite relation associated with $\mathcal{R}$ is denoted by $\to_{\mathcal{R}}$.

For every term t, let $\Diamond t$ be a "tree with shared variables" representing t, i.e., $\Diamond t$ is a term graph such that (1) $term(\Diamond t) = t$, (2) $indegree(v) \leq 1$ for each node v with $l_{\Diamond t}(v) \notin X$,[1] and (3) $v = v'$ for all nodes v,v' with $l_{\Diamond t}(v) = l_{\Diamond t}(v') \in X$.

For every term graph G, let $\underline{G}$ be the hypergraph that is obtained from G by removing all edges labelled with variables.

Given two hypergraphs G and H, a *hypergraph morphism* $g: G \to H$ is a pair of mappings $\langle g_V: V_G \to V_H, g_E: E_G \to E_H \rangle$ that preserve sources, targets, and labels, i.e., $s_H \circ g_E = g_V \circ s_G$, $t_H \circ g_E = g_V^* \circ t_G$, and $l_H \circ g_E = l_G$.[2]

What follows are constructions of the hypergraph and term graph rewrite steps used in this paper. Exact definitions based on hypergraph pushouts are given in [16].

Let G and H be hypergraphs. Then there is an *evaluation step* from G to H, denoted by $G \Rightarrow_{\mathcal{E}} H$, if there is a rewrite rule $l \to r$ in $\mathcal{R}$ and a hypergraph morphism $g: \underline{\Diamond l} \to G$ (determining the *redex* $g(\underline{\Diamond l})$) such that H is isomorphic[3] to the hypergraph constructed from G as follows:
(1) Remove $g(e)$, where e is the edge with source $root_{\Diamond l}$, yielding a hypergraph G'.
(2) Build the disjoint union $G' + \underline{\Diamond r}$ and
(2.1) identify $g(root_{\Diamond l})$ with $root_{\Diamond r}$,
(2.2) for each pair $\langle v, v' \rangle \in V_{\Diamond l} \times V_{\Diamond r}$ with $l_{\Diamond l}(v) = l_{\Diamond r}(v') \in X$, identify $g(v)$ with v'.

To turn $\Rightarrow_{\mathcal{E}}$ into a relation on term graphs, evaluation steps are completed by a garbage collection phase. Given two term graphs G and J, write $G \Rrightarrow_{\mathcal{E}} J$ if there is an evaluation step $G \Rightarrow_{\mathcal{E}} H$ such that J is obtained from H as follows:
(3) Remove all edges e (and their source nodes) satisfying $root_G \not\geq_H s_H(e)$.

Figure 1 shows an evaluation step (with garbage collection) by the rewrite rule $x \times (y+z) \to (x \times y)+(x \times z)$. Note that the morphism locating $\underline{\Diamond x \times (y+z)}$ identifies the nodes representing y and z, and that the +-edge with shared target nodes is removed by garbage collection.

Besides evaluation steps, term graphs are manipulated by so-called folding steps which enhance the degree of sharing. This is to make term graph rewriting a complete proof method for first-order equations. Moreover, in certain cases folding steps speed up the evaluation process considerably (see [7] for an example in which an exponential number of steps is reduced to a linear number).

A *folding step* $G \Rightarrow_{\mathcal{F}} \widetilde{G}$ between two hypergraphs G and $\widetilde{G}$ is constructed by identifying two distinct edges e and e' in G that satisfy $l_G(e) = l_G(e')$ and $t_G(e) = t_G(e')$. Note that if G is a term graph, then so is $\widetilde{G}$ and $term(G) = term(\widetilde{G})$. An example of a folding step is shown in Figure 1.

This paper deals with arbitrary sequences of evaluation and folding steps. Given two hypergraphs G and H, write $G \Rightarrow_{\mathcal{R}} H$ if $G \Rightarrow_{\mathcal{E}} H$ or $G \Rightarrow_{\mathcal{F}} H$. For term graphs G and H, write $G \Rrightarrow_{\mathcal{R}} H$ if $G \Rrightarrow_{\mathcal{E}} H$ or $G \Rightarrow_{\mathcal{F}} H$.

---

[1] $indegree(v)$ is the number of occurrences of v in the target strings of all $\Diamond t$-edges.

[2] Given a mapping $f: A \to B$, $f^*: A^* \to B^*$ sends a string $a_1 \ldots a_n$ to $f(a_1) \ldots f(a_n)$.

[3] A hypergraph morphism $g: G \to H$ is an *isomorphism* if g_V and g_E are bijective. In this case G and H are *isomorphic*.

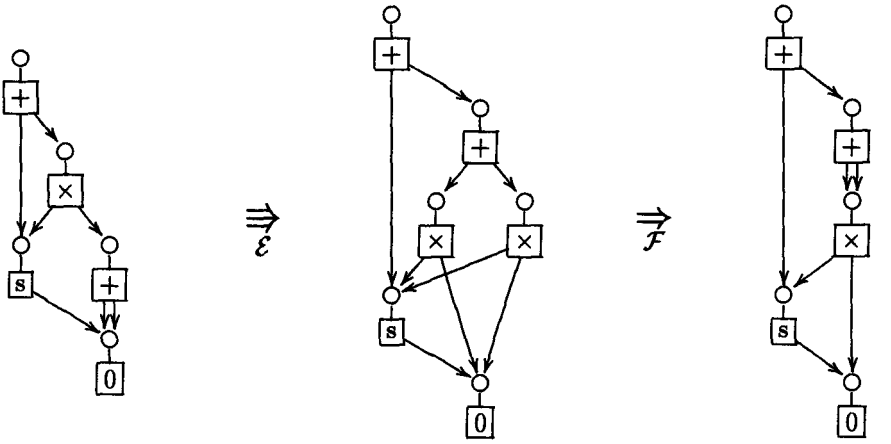

Figure 1: An evaluation step followed by a folding step

For every step $G \Rightarrow_{\mathcal{R}} H$ there is function $track_{G \Rightarrow H}: V_G \to V_H$ which sends each node in G to its "descendant" in H (see [16] for the precise definition based on pushouts). Track functions are extended to rewrite sequences by composing the track functions of the constituting rewrite steps.

Given a binary relation $\to$ on a set A, $\to^*$ and $\leftrightarrow^*$ denote the transitive-reflexive and symmetric-transitive-reflexive closures of $\to$. The relation $\to$ is *(locally) confluent* if for all a, b, c with $b \leftarrow^* a \to^* c$ ($b \leftarrow a \to c$) there is some d such that $b \to^* d \leftarrow^* c$. The relation $\to$ is *terminating* if there is no infinite sequence $a_1 \to a_2 \to a_3 \to \ldots$. An element a in A is a *normal form* if there is no b such that $a \to b$. Element a *has a normal form* if $a \to^* b$ for some normal form b.

3 Critical Pairs and Confluence

The goal of the following considerations is to give a sufficient condition for local confluence of term graph rewriting in form of a *Critical Pair Lemma*, analogously to the well-known result established by Knuth and Bendix [11] and Huet [8]. The idea is to infer local confluence of $\Rightarrow_{\mathcal{R}}$ from the confluence of so-called critical pairs, being certain divergent steps $U_1 \Leftarrow_{\mathcal{R}} \underline{S} \Rightarrow_{\mathcal{R}} U_2$ in which $\underline{S}$ represents a "critical overlap" of the involved redexes.

As a prerequisite for the Critical Pair Lemma, one has to show that $\Rightarrow_{\mathcal{R}}$ is already locally confluent if for each two steps of the form $H_1 \Leftarrow_{\mathcal{E}} G \Rightarrow_{\mathcal{E}} H_2$ there is a term graph M with $H_1 \Rightarrow^*_{\mathcal{R}} M \Leftarrow^*_{\mathcal{R}} H_2$. Since folding is confluent, this amounts to showing that for each divergent situation $H_1 \Leftarrow_{\mathcal{E}} G \Rightarrow_{\mathcal{F}} H_2$ there is also such an M. The crucial case is given when the edge removed in

$G \Rightarrow\!\!\!\!\!\!\cdot_{\mathcal{E}} H_1$ is one of the two edges identified in $G \Rightarrow_{\mathcal{F}} H_2$; constructing M in this constellation makes up the main part of the proof of the following theorem.

Theorem 3.1 *The relation $\Rightarrow_{\mathcal{R}}$ is locally confluent if and only if for all steps $H_1 \Leftarrow_{\mathcal{E}} G \Rightarrow\!\!\!\!\!\!\cdot_{\mathcal{E}} H_2$ there is a term graph M such that $H_1 \Rightarrow^*_{\mathcal{R}} M \Leftarrow^*_{\mathcal{R}} H_2$.*

By this result, testing term graph rewriting for local confluence reduces to the problem of checking that every pair of divergent evaluation steps is confluent under $\Rightarrow_{\mathcal{R}}$. The Critical Pair Lemma established below provides a sufficient condition for the latter showing that it suffices to consider only those divergent steps $U_1 \Leftarrow_{\mathcal{E}} \underline{S} \Rightarrow\!\!\!\!\!\!\cdot_{\mathcal{E}} U_2$ for which $\underline{S}$ results from superposing the redexes. The requirement is then that there are derivations $U_1 \Rightarrow^*_{\mathcal{R}} X_1$ and $U_2 \Rightarrow^*_{\mathcal{R}} X_2$ such that X_1 and X_2 are isomorphic up to certain garbage by an isomorphism that is compatible with the track functions of the derivations $\underline{S} \Rightarrow_{\mathcal{E}} U_1 \Rightarrow^*_{\mathcal{R}} X_1$ and $\underline{S} \Rightarrow_{\mathcal{E}} U_2 \Rightarrow^*_{\mathcal{R}} X_2$.

When evaluation steps are embedded into a (graph-)context, edges may be attached to nodes that become garbage. Therefore critical pairs have to be defined for evaluation steps without garbage collection. In the following definition, the first condition puts a bound on the size of hypergraphs in critical pairs if the given set $\mathcal{R}$ of rewrite rules is finite, while the second condition expresses that redexes overlap in a critical way.

Definition 3.2 (critical pair) Let S be a term graph and $\underline{S} \Rightarrow\!\!\!\!\!\!\cdot_{\mathcal{E}} U_i$ be an evaluation step by a rewrite rule $l_i \to r_i$ and a hypergraph morphism $g_i\colon \underline{\Diamond l_i} \to \underline{S}$, for $i = 1, 2$. Then $U_1 \Leftarrow_{\mathcal{E}} \underline{S} \Rightarrow\!\!\!\!\!\!\cdot_{\mathcal{E}} U_2$ is a *critical pair* (over $\mathcal{R}$) if

(1) $\underline{S} = g_1(\underline{\Diamond l_1}) \cup g_2(\underline{\Diamond l_2})$ and

(2) $g_1(e_1) \in g_2(\underline{\Diamond l_2})$ or $g_2(e_2) \in g_1(\underline{\Diamond l_1})$, where e_i is the edge outgoing from $root_{\Diamond l_i}$ for $i = 1, 2$.

Moreover, $g_1 \neq g_2$ is required for the case $(l_1 \to r_1) = (l_2 \to r_2)$.

In what follows, critical pairs that differ only by renaming of nodes and edges are not distinguished. As a consequence, only finitely many critical pairs need to be considered whenever $\mathcal{R}$ is finite.

Example 3.3 The rewrite rules

$$\begin{array}{rcl} \mathrm{g(f(a,x))} & \to & \mathrm{g(b)} \\ \mathrm{f(x,y)} & \to & \mathrm{c} \end{array}$$

(with x,y being variables) give rise to the two critical pairs shown in Figure 2 (the nodes are numbered to indicate the track functions).

Given a hypergraph derivation $G \Rightarrow^*_{\mathcal{R}} H$, let $Track(G \Rightarrow^*_{\mathcal{R}} H)$ be the subhypergraph of H with node set $\{w \in V_H \mid track_{G \Rightarrow^* H}(v) \geq_H w \text{ for some } v \in V_G\}$ and edge set $\{e \in E_H \mid s_H(e) \in V_{Track(G \Rightarrow^*_{\mathcal{R}} H)}\}$.

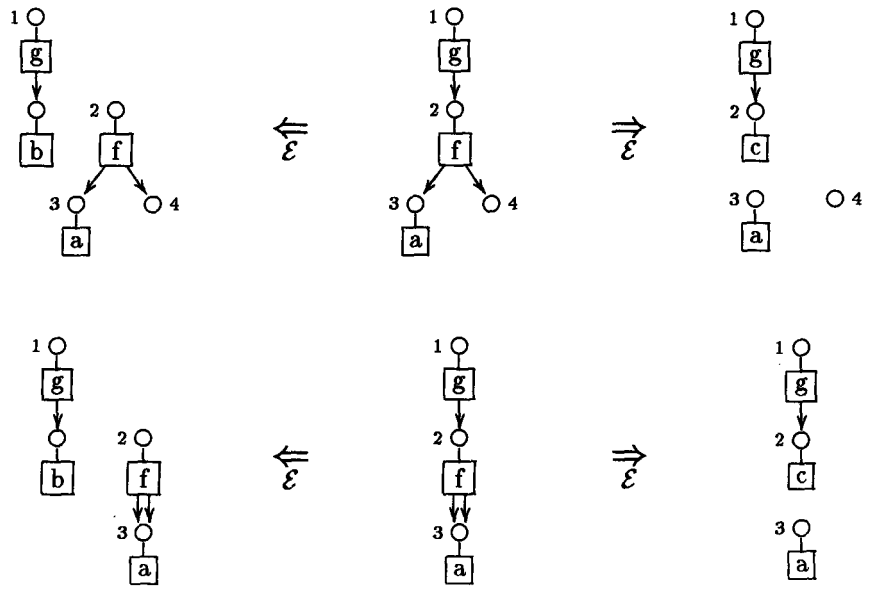

Figure 2: Two critical pairs

Definition 3.4 (joinability) A critical pair $U_1 \Leftarrow_{\mathcal{E}} \underline{S} \Rightarrow_{\mathcal{E}} U_2$ is *joinable* if there are derivations $U_1 \Rightarrow_{\mathcal{R}}^* X_1$ and $U_2 \Rightarrow_{\mathcal{R}}^* X_2$ such that there is an isomorphism $iso: Track(\underline{S} \Rightarrow_{\mathcal{E}} U_1 \Rightarrow_{\mathcal{R}}^* X_1) \rightarrow Track(\underline{S} \Rightarrow_{\mathcal{E}} U_2 \Rightarrow_{\mathcal{R}}^* X_2)$ satisfying $iso_V(track_{\underline{S} \Rightarrow U_1 \Rightarrow^* X_1}(v)) = track_{\underline{S} \Rightarrow U_2 \Rightarrow^* X_2}(v)$ for each $v \in V_{\underline{S}}$.

Note that the joining derivations $U_i \Rightarrow_{\mathcal{R}}^* X_i$ are allowed to contain folding steps and that garbage in X_1 and X_2 is ignored as far as it consists of items not reachable from the descendant of any node in $\underline{S}$. Requiring that X_1 and X_2 are isomorphic would be too restrictive, as can be seen in the following example.

Example 3.5 After adding the rewrite rule g(b) $\rightarrow$ g(c) to the two rules of Example 3.3, the second critical pair in Figure 2 is joinable by the derivation shown in Figure 3. Observe that the resulting hypergraph contains garbage (the constant b) which does not occur in the right-hand hypergraph of the critical pair. The first critical pair in Figure 2 is joinable by a similar derivation.

Lemma 3.6 (Critical Pair Lemma) *If all critical pairs over $\mathcal{R}$ are joinable, then $\Rightarrow_{\mathcal{R}}$ is locally confluent.*

The proof of the Critical Pair Lemma in [16] is based on two results from graph grammar theory stating a commutation property for independent rewrite steps (where independence corresponds to absence of critical pairs) and the possibility of embedding derivations into context.

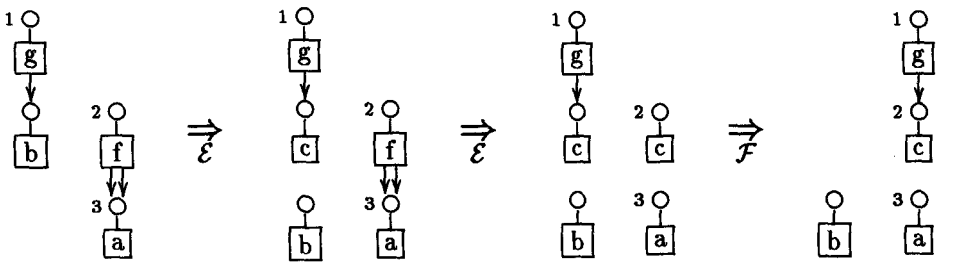

Figure 3: A joining derivation for Example 3.3

The Critical Pair Lemma allows to characterize confluence in the presence of termination, analogously to the well-known result for term rewriting [11].

Theorem 3.7 *Suppose that $\Rightarrow_{\mathcal{R}}$ is terminating. Then $\Rightarrow_{\mathcal{R}}$ is confluent if and only if all critical pairs over $\mathcal{R}$ are joinable.*

One direction of this result follows directly from the Critical Pair Lemma by the fact that local confluence together with termination implies confluence [8]. The proof of the converse direction exploits that $\Rightarrow_{\mathcal{R}}$ is terminating if and only if $\Rightarrow_{\mathcal{R}}$ is, and that "fully collapsed" term graphs (i.e. normal forms of $\Rightarrow_{\mathcal{F}}$) are—up to isomorphism—uniquely determined by the terms they represent.

Example 3.8 Let $\mathcal{R}$ be the following system:

$$\begin{array}{rcl} \mathrm{f(x)} & \to & \mathrm{g(x,x)} \\ \mathrm{a} & \to & \mathrm{b} \\ \mathrm{g(a,b)} & \to & \mathrm{f(a)} \end{array}$$

To see that $\Rightarrow_{\mathcal{R}}$ is terminating, observe that no evaluation or folding step increases the number of g-labelled edges with distinct target nodes. Therefore an infinite sequence $G_1 \Rightarrow_{\mathcal{R}} G_2 \Rightarrow_{\mathcal{R}} \ldots$ had to contain a term graph G_k such that the number of these edges remains constant in all G_n with $n \geq k$. So the sequence $G_k \Rightarrow_{\mathcal{R}} G_{k+1} \Rightarrow_{\mathcal{R}} \ldots$ could not contain evaluation steps with the third rewrite rule. But an infinite sequence without these steps is impossible (assign to a term graph G the sum $|l_G^{-1}(\mathrm{f})| + |l_G^{-1}(\mathrm{a})| + |E_G|$ which decreases in each step). Thus, as there is only one critical pair which is easily shown to be joinable, $\Rightarrow_{\mathcal{R}}$ is confluent by Theorem 3.7.

From the proof of Theorem 3.7 a test can be derived which—in the presence of termination—decides whether $\Rightarrow_{\mathcal{R}}$ is confluent or not.

Theorem 3.9 *The procedure in Figure 4 solves the following problem:*

Instance: *A term rewriting system $\mathcal{R}$ with finitely many rules such that $\Rightarrow_{\mathcal{R}}$ is terminating.*

Question: *Is $\Rightarrow_{\mathcal{R}}$ confluent?*

input: a term rewriting system $\mathcal{R}$ with finitely many rules such that $\Rightarrow_{\mathcal{R}}$ is terminating

begin
  for each critical pair $U_1 \Leftarrow_\varepsilon \underline{S} \Rightarrow_\varepsilon U_2$ **do**
    extend $\underline{S}$ to S by appending variable-edges, and construct extended evaluation steps $\widehat{U}_1 \Leftarrow_\varepsilon S \Rightarrow_\varepsilon \widehat{U}_2$;
    starting with $\widehat{U}_1$ and $\widehat{U}_2$, perform evaluation and folding steps as long as possible to obtain derivations $\widehat{U}_1 \Rightarrow^*_{\mathcal{R}} \widehat{X}_1$ and $\widehat{U}_2 \Rightarrow^*_{\mathcal{R}} \widehat{X}_2$ such that $\widehat{X}_1$ and $\widehat{X}_2$ are normal forms;
    $V := V_S$;
    repeat
      choose some node v in V;
      if $term_{\widehat{X}_1}(track_{S \Rightarrow \widehat{U}_1 \Rightarrow^* \widehat{X}_1}(v)) = term_{\widehat{X}_2}(track_{S \Rightarrow \widehat{U}_2 \Rightarrow^* \widehat{X}_2}(v))$
      then $V := V - \{v\}$ **else return**("$\Rightarrow_{\mathcal{R}}$ is not confluent")
    until $V = \emptyset$
  endfor;
  write("$\Rightarrow_{\mathcal{R}}$ is confluent")
end

Figure 4: Decision procedure for confluence

4 The Relation to Term Rewriting

This section clarifies the relation between term graph and term rewriting. In particular, the class of systems $\mathcal{R}$ for which $\Rightarrow_{\mathcal{R}}$ is confluent and terminating is shown to be a proper superclass of those systems that are confluent and terminating under $\rightarrow_{\mathcal{R}}$. Moreover, it turns out that the decision procedure in Figure 4 yields a confluence test for term rewriting that extends the classical one of Knuth and Bendix.

By the following principal result, term graph rewriting is sound and complete for equational deduction in the same sense as term rewriting is.

Theorem 4.1 (Completeness Theorem [15]) *For all term graphs G and H,*

$$G \Leftrightarrow^*_{\mathcal{R}} H \text{ if and only if } term(G) \leftrightarrow^*_{\mathcal{R}} term(H).$$

With the well-known equivalence of confluence and the Church-Rosser property [8] one obtains the following corollary.

Corollary 4.2 *Suppose that $\Rightarrow_{\mathcal{R}}$ is confluent. Then for all term graphs G,H, $term(G) \leftrightarrow^*_{\mathcal{R}} term(H)$ if and only if $G \Rightarrow^*_{\mathcal{R}} M \Leftarrow^*_{\mathcal{R}} H$ for some term graph M.*

As a consequence, the relation $\leftrightarrow^*_{\mathcal{R}}$ (which coincides with equality in the models of $\mathcal{R}$) is decidable (for finite $\mathcal{R}$) whenever $\Rightarrow_{\mathcal{R}}$ is terminating: Given terms t

and u, choose term graph representations T and U and perform evaluation and folding steps as long as possible, obtaining normal forms T' and U'; then $t \leftrightarrow^*_{\mathcal{R}} u$ if and only if T' and U' are isomorphic.

From Corollary 4.2 and soundness of term graph rewriting ($G \Rrightarrow_{\mathcal{R}} H$ implies $term(G) \rightarrow^*_{\mathcal{R}} term(H)$) one obtains the following relationship.

Theorem 4.3 ([15]) *If $\Rrightarrow_{\mathcal{R}}$ is confluent, then so is $\rightarrow_{\mathcal{R}}$.*

The converse of this result does not hold, a counterexample is given in [15, 16]. Despite this relationship, term graph rewriting has unique normal forms if and only if term rewriting has. Recall that a binary relation $\rightarrow$ *has unique normal forms* if for all normal forms a and b, $a \leftrightarrow^* b$ implies $a = b$.

Theorem 4.4 *The relation $\Rrightarrow_{\mathcal{R}}$ has unique normal forms if and only if $\rightarrow_{\mathcal{R}}$ has.*

The proof rests on the Completeness Theorem 4.1, the fact that $\Rrightarrow_{\mathcal{R}}$-normal forms represent term normal forms, and the uniqueness of fully collapsed term graphs. As a consequence of this characterization, terminating term graph rewriting is confluent whenever term rewriting is.

Corollary 4.5 *Suppose that $\Rrightarrow_{\mathcal{R}}$ is terminating. Then $\Rrightarrow_{\mathcal{R}}$ is confluent if and only if $\rightarrow_{\mathcal{R}}$ is.*

Thus, the decision procedure in Figure 4 can be used as a confluence test for term rewriting. Since termination of $\rightarrow_{\mathcal{R}}$ carries over to $\Rrightarrow_{\mathcal{R}}$ but not vice versa, the procedure applies to all terminating and also to certain non-terminating term rewriting systems. For instance, in Example 3.8 it is shown that $\Rrightarrow_{\mathcal{R}}$ is confluent and terminating for the system $\mathcal{R}$ given there. So the procedure discovers that $\rightarrow_{\mathcal{R}}$ is confluent. However, $\rightarrow_{\mathcal{R}}$ is non-terminating because of the infinite sequence

$$\mathrm{f(a)} \rightarrow_{\mathcal{R}} \mathrm{g(a,a)} \rightarrow_{\mathcal{R}} \mathrm{g(a,b)} \rightarrow_{\mathcal{R}} \mathrm{f(a)} \rightarrow_{\mathcal{R}} \ldots$$

and hence the confluence test of Knuth and Bendix [11] for terminating term rewriting systems does not apply. Their test reduces the terms in critical pairs (in the sense of term rewriting) to normal form and checks syntactical equality. But, in general, the joinability of all critical pairs does not guarantee confluence if $\rightarrow_{\mathcal{R}}$ is non-terminating. (Observe also that in this example normalization may fail because the term f(a) occurs in the only critical pair.)

As term graph rewriting is terminating whenever term rewriting is, Corollary 4.5 yields also the following relationship.

Corollary 4.6 *If $\rightarrow_{\mathcal{R}}$ is confluent and terminating, then so is $\Rrightarrow_{\mathcal{R}}$.*

The converse does not hold, since $\rightarrow_{\mathcal{R}}$ needs not be terminating when $\Rrightarrow_{\mathcal{R}}$ is (see the above example). So confluent and terminating term rewriting systems form a proper subclass of the systems that enjoy this property under term graph rewriting. As a result, the term graph decision procedure for $\leftrightarrow^*_{\mathcal{R}}$ described after Corollary 4.2 terminates for more systems than the corresponding procedure by term rewriting.

5 Concluding Remarks

As pointed out to the author by H. Comon, it seems evident that the above results allow improved confluence criteria for *equational term rewriting* which comprises applications of both rules and equations. Previous criteria developed by Padawitz [12] and Jouannaud, Kirchner and Remy [9] are based on so-called parallel critical pairs in order to avoid the restriction that rules have to be right-linear. Using term graph rewriting instead of term rewriting should yield simplied conditions for an equational theory to be decidable.

Another line of research related to the present approach deals with critical pairs over general graph rewrite rules [17]. This setting is more general in that rules operate on arbitrary (hyper-)graphs, and different in so far as rewrite steps do not include garbage collection. Here the joinability condition needed for critical pairs is not necessary for confluence, even in the presence of termination. In contrast to the situation for term graph rewriting, confluence of terminating graph rewrite systems turns out to be undecidable in general.

References

[1] H. Barendregt, M. van Eekelen, J. Glauert, R. Kennaway, R. Plasmeijer, and R. Sleep. Term graph rewriting. In *Proc. Parallel Architectures and Languages Europe*, pages 141–158. Springer Lecture Notes in Computer Science 259, 1987.

[2] A. Corradini and F. Rossi. Hyperedge replacement jungle rewriting for term rewriting systems and logic programming. *Theoretical Computer Science*, 109:7–48, 1993.

[3] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science,* volume B, chapter 6. Elsevier, 1990.

[4] W. M. Farmer and R. J. Watro. Redex capturing in term graph rewriting. *International Journal on Foundations of Computer Science*, 1(4), 1990.

[5] J. H. Fasel and R. M. Keller, editors. *Graph Reduction.* Springer Lecture Notes in Computer Science 279, 1987.

[6] J. Goguen, C. Kirchner, and J. Meseguer. Concurrent term rewriting as a model of computation. In *Proc. Graph Reduction*, pages 53–93. Springer Lecture Notes in Computer Science 279, 1987.

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

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

[9] J.-P. Jouannaud, H. Kirchner, and J.-L. Rémy. Church-rosser properties of weakly terminating term rewriting systems. In *Proc. International Joint Conference on Artificial Intelligence '83*, pages 909–915, 1983.

[10] J. W. Klop. Term rewriting systems. In S. Abramsky, D. M. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 1–116. Oxford University Press, 1992.

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

[12] P. Padawitz. Equational data type specifications and recursive program schemes. In *Proc. Formal Descriptions of Programming Concepts-II*, pages 305–328. North-Holland, 1983.

[13] S. L. Peyton Jones. *The Implementation of Functional Programming Languages.* Prentice-Hall, 1987.

[14] D. A. Plaisted. Equational reasoning and term rewriting systems. In D. M. Gabbay and J. Siekmann, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1. Oxford University Press, 1993.

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

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

[17] D. Plump. Hypergraph rewriting: Critical pairs and undecidability of confluence. In [18], chapter 15.

[18] R. Sleep, R. Plasmeijer, and M. van Eekelen, editors. *Term Graph Rewriting: Theory and Practice.* John Wiley, 1993.