Implementing Term Rewriting by Graph Reduction: Termination of Combined Systems *

Detlef Plump
Fachbereich Mathematik und Informatik
Universität Bremen, Postfach 33 04 40
D-2800 Bremen 33

Abstract

It is well known that the disjoint union of terminating term rewriting systems does not yield a terminating system in general. We show that this undesirable phenomenon vanishes if one implements term rewriting by graph reduction: given two terminating term rewrite systems $\mathcal{R}_0$ and $\mathcal{R}_1$, the graph reduction system implementing $\mathcal{R}_0 + \mathcal{R}_1$ is terminating. In fact, we prove the stronger result that the graph reduction system for the union $\mathcal{R}_0 \cup \mathcal{R}_1$ is terminating provided that the left-hand sides of $\mathcal{R}_i$ have no common function symbols with the right-hand sides of $\mathcal{R}_{1-i}$ $(i = 0, 1)$.

The implementation is complete in the sense that it computes a normal form for each term over the signature of $\mathcal{R}_0 \cup \mathcal{R}_1$.

1 Introduction

The operational semantics of algebraic specification languages are usually based on term rewriting (see, e.g., [BCV 85], [BHK 89], [EM 85], [FGJM 85], [GH 86]). In this context, confluence and termination are particularly relevant properties of term rewriting systems. Hence, when specifications are structured as combinations of smaller subspecifications, the question arises whether these properties are preserved by a given combination mechanism.

Recently, research in this direction has been started by considering the disjoint union $\mathcal{R}_0 + \mathcal{R}_1$ of two term rewriting systems $\mathcal{R}_0$ and $\mathcal{R}_1$: the rule set of $\mathcal{R}_0 + \mathcal{R}_1$ is the union of the rules of $\mathcal{R}_0$ and $\mathcal{R}_1$ where the function symbols occurring in $\mathcal{R}_0$ and $\mathcal{R}_1$ are disjoint (or are made disjoint by renaming). Toyama [Toy 87a] proves that $\mathcal{R}_0 + \mathcal{R}_1$ is confluent

*Work supported by ESPRIT project #390, *PROSPECTRA*, and by ESPRIT Basic Research Working Group #3264, *COMPASS*.

if both components are confluent. However, the corresponding result for termination does not hold, as the following counterexample [Toy 87b] shows:

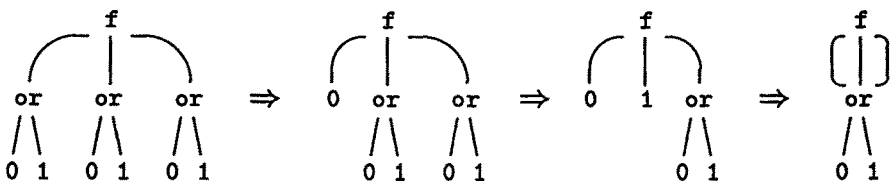
$$\begin{aligned}\mathcal{R}_0: &\ \mathtt{f(0,1,x)} \to \mathtt{f(x,x,x)}\\ \mathcal{R}_1: &\ \mathtt{or(x,y)} \to \mathtt{x},\\ &\ \mathtt{or(x,y)} \to \mathtt{y}\end{aligned}$$

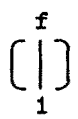
$\mathcal{R}_0$ and $\mathcal{R}_1$ are terminating, while $\mathcal{R}_0 + \mathcal{R}_1$ is not: `f(or(0,1),or(0,1),or(0,1))` reduces in three steps to itself. Examples given by Barendregt and Klop (see [Toy 87b]) and Drosten [Dro 89] show that not even the disjoint union of convergent (i.e. confluent and terminating) systems is guaranteed to terminate.

Sufficient conditions for the termination of the disjoint union of terminating term rewrite systems are given by Ganzinger and Giegerich [GG 87], Rusinowitch [Rus 87], Drosten [Dro 89], Middeldorp [Mid 89a], and Toyama, Klop and Barendregt [TKB 89]. These authors impose various restrictions upon the involved rewrite systems. (See also [Mid 89b] for results on conditional term rewriting systems.)

In this paper we show that these restrictions can be dropped if term rewriting is implemented by graph reduction. More precisely, we use the framework of ***jungle evaluation*** [HP 88] where terms are represented by (hyper-)graphs and term rewrite rules are translated into graph reduction rules such that graph reduction steps correspond to (parallel) term rewriting steps. Let $\mathcal{G}(\mathcal{R})$ denote the graph reduction system corresponding to a term rewriting system $\mathcal{R}$. Then $\mathcal{G}(\mathcal{R})$ implements $\mathcal{R}$ in the following sense: if a term t is represented by a graph G and $G \overset{*}{\underset{\mathcal{G}(\mathcal{R})}{\Longrightarrow}} H$ is a reduction with H in normal form (i.e., no rule in $\mathcal{G}(\mathcal{R})$ is applicable), then H represents a normal form of t.

In [HP 88] it is shown that $\mathcal{G}(\mathcal{R})$ is terminating if $\mathcal{R}$ is terminating. However, the converse does not hold and we will see that $\mathcal{G}(\mathcal{R}_0 + \mathcal{R}_1)$ is terminating whenever $\mathcal{R}_0$ and $\mathcal{R}_1$ are terminating, no matter whether $\mathcal{R}_0 + \mathcal{R}_1$ is terminating or not. As an example, consider the term rewriting systems $\mathcal{R}_0$ and $\mathcal{R}_1$ shown above. Starting with the tree representation of `f(or(0,1),or(0,1),or(0,1))` we get the following graph reduction steps[1]:



At this stage only one further step is possible: either  or  is computed, both representing normal forms of the input term.

The crucial point for the termination of the above reduction is the application of the graph reduction rule corresponding to $\mathtt{f(0,1,x)} \to \mathtt{f(x,x,x)}$. While two copies of the

[1] Garbage arising from reduction steps is not depicted.

subterm matched by x are produced in the term case, graph reduction only introduces two additional pointers to this argument. As a consequence, the three occurrences of `or(0,1)` in `f(or(0,1),or(0,1),or(0,1))` are represented only once and cannot be rewritten independently. Actually, Rusinowitch [Rus 87] shows that the disjoint union of term rewriting systems preserves termination if no such duplicating rules are present.

The disjoint union is a rather restrictive combination mechanism for term rewriting systems, in view of the disjointness requirement. This requirement may be relaxed by considering the union $\mathcal{R}_0 \cup \mathcal{R}_1$ of two term rewriting systems. Sufficient conditions for the termination of $\mathcal{R}_0 \cup \mathcal{R}_1$ are given by Bachmair and Dershowitz [BD 86]. They show that the termination of $\mathcal{R}_0$ and $\mathcal{R}_1$ implies the termination of $\mathcal{R}_0 \cup \mathcal{R}_1$ provided that $\mathcal{R}_0$ is left-linear, $\mathcal{R}_1$ is right-linear, and there is no overlap between the left-hand sides of $\mathcal{R}_0$ and the right-hand sides of $\mathcal{R}_1$. (The overlap condition implies that no right-hand side in $\mathcal{R}_1$ is a single variable.)

In section 3 we consider the union $\mathcal{R}_0 \cup \mathcal{R}_1$ where the left-hand sides of $\mathcal{R}_i$ have no common function symbols with the right-hand sides of $\mathcal{R}_{1-i}$ $(i = 0, 1)$. Our main result states that in this case the termination of $\mathcal{R}_0$ and $\mathcal{R}_1$ guarantees the termination of $\mathcal{G}(\mathcal{R}_0 \cup \mathcal{R}_1)$ without further restrictions. In fact, the termination of $\mathcal{R}_0$ and $\mathcal{R}_1$ is not necessary for the termination of $\mathcal{G}(\mathcal{R}_0 \cup \mathcal{R}_1)$: it suffices that $\mathcal{G}(\mathcal{R}_0)$ and $\mathcal{G}(\mathcal{R}_1)$ are terminating.

This result shows that terminating graph reduction implementations can, with respect to termination, be safely combined as long as the left- and right-hand sides of the corresponding term rewrite systems are disjoint in the described manner. The resulting system is guaranteed to terminate although the underlying term rewrite system may fail to terminate. In particular, this result together with the termination theorem from [HP 88] (see section 2 below) shows that the union $\bigcup_{i=1}^{n} \mathcal{R}_i$ of a family of terminating term rewrite systems $(\mathcal{R}_i)_{i=1,\ldots,n}$ has a terminating graph reduction implementation, provided these systems satisfy the above disjointness condition pairwise.

2 Jungle Evaluation

This section provides an introduction to jungle evaluation ([HKP 88], [HP 88]), being the setting in which our result is proved. We assume that the reader is familiar with basic notions of term rewriting (see for example [DJ 90], [Klo 90]).

We pursue a many-sorted approach. Let $SIG = (S, F)$ be a signature, that is, S is a set of sorts and F is a set of function symbols such that each $f \in F$ has a string of argument sorts $s_1 \ldots s_n \in S^*$ and a result sort $s \in S$. A function symbol f is written $f : s_1 \ldots s_n \to s$ when the argument and result sorts matter.

A *hypergraph* $G = (V_G, E_G, s_G, t_G, l_G, m_G)$ over SIG consists of a finite set V_G of nodes, a finite set E_G of hyperedges (or edges for short), two mappings $s_G : E_G \to V_G^*$ and $t_G : E_G \to V_G^*$, assigning a string of source nodes and a string of target nodes to each hyperedge, and two mappings $l_G : V_G \to S$ and $m_G : E_G \to F$, labeling nodes with sorts and hyperedges with function symbols.

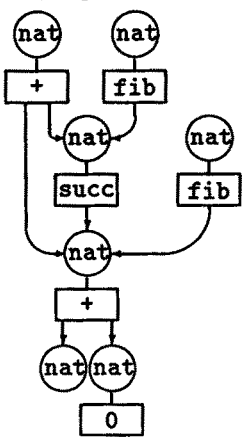
A hypergraph G over SIG is a *jungle* if

- the labeling of G is compatible with SIG, that is, for each $e \in E_G$, $m_G(e) = f : s_1 \dots s_n \to s$ implies $l^*_G(s_G(e)) = s$ and $l^*_G(t_G(e)) = s_1 \dots s_n$, [2]
- $outdegree_G(v) \leq 1$ for each $v \in V_G$,
- G is acyclic.

Example. Assume that SIG contains a sort nat and function symbols

$$\begin{aligned} &0 : \to \mathtt{nat}, \\ &\mathtt{succ}, \mathtt{fib} : \mathtt{nat} \to \mathtt{nat}, \\ &+ : \mathtt{nat}\ \mathtt{nat} \to \mathtt{nat}. \end{aligned}$$

Then the following hypergraph is a jungle over SIG.



Nodes are drawn as circles and hyperedges as boxes, both with their labels written inside. A line without arrow-head connects a hyperedge with its unique source node, while arrows point to the target nodes (if there are any). The arrows are arranged from left to right in the order given by the target mapping. □

When SIG-terms shall be represented by jungles, nodes without outgoing hyperedge serve as variables. The set of all such nodes in a jungle G is denoted by VAR_G. (Note that as a consequence of this definition each variable occurs only once in a jungle.) The term represented by a node v is then defined by

$$term_G(v) = \begin{cases} v & \text{if } v \in VAR_G, \\ m_G(e)term^*_G(t_G(e)) & \text{otherwise, where } e \text{ is the unique} \\ & \text{hyperedge with } s_G(e) = v. \end{cases}$$

For instance, if v is the left one of the two topmost nodes in the above example, then $term_G(v) = (\mathtt{x}+0)+\mathtt{succ}(\mathtt{x}+0)$ (in infix notation) where x is the unique variable node in G.

[2] Given a mapping $g : A \to B$, the extension $g^* : A^* \to B^*$ is defined by $g^*(\lambda) = \lambda$ and $g^*(aw) = g(a)g^*(w)$ for all $a \in A$, $w \in A^*$. Here λ denotes the empty string.

For defining reduction rules and reduction steps we need structure preserving mappings between jungles. A *jungle morphism* $g : G \to H$ consists of two mappings $g_V : V_G \to V_H$ and $g_E : E_G \to E_H$ which preserve sources, targets, and labels, that is, $s_H \circ g_E = g_V \circ s_G$, $t_H \circ g_E = g_V^* \circ t_G$, $l_H \circ g_V = l_G$, and $m_H \circ g_E = m_G$.

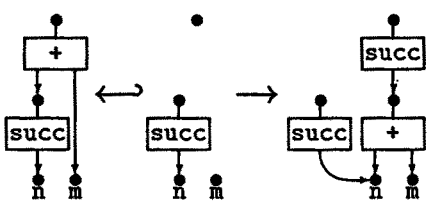
Now we are ready to translate term rewrite rules into *evaluation rules*. For motivations and further details, however, we refer to [HP 88].

Let $\mathcal{R}$ be a term rewriting system over SIG, i.e., each rule $l \to r$ in $\mathcal{R}$ consists of two SIG-terms l and r of equal sort such that l is not a variable and all variables in r occur already in l. Given a rule $l \to r$ from $\mathcal{R}$, the corresponding evaluation rule $(L \hookleftarrow K \xrightarrow{b} R)$ consists of three jungles L,K,R and jungle morphisms $K \hookrightarrow L$ and $b : K \to R$ such that:

- L is a variable-collapsed tree[3] with $term_L(root_L) = l$.
- K is the subjungle of L obtained by removing the edge outgoing from $root_L$.
- R is constructed from K as follows: if r is a variable, then $root_L$ is identified with the node r; otherwise, R is the disjoint union of K and a variable-collapsed tree R' with $term_{R'}(root_{R'}) = r$ where $root_{R'}$ is identified with $root_L$ and each $x \in VAR_{R'}$ is identified with its counterpart in VAR_K.
- $K \hookrightarrow L$ and $b : K \to R$ are inclusions with the possible exception that b identifies $root_L$ with some variable.

Remark. For simplicity we choose R to be as little collapsed as possible. Other forms of R may be used as well, including the one where R is "fully collapsed" (cf. [HP 88]).

Example. The following picture shows the evaluation rule for the rewrite rule $\mathtt{succ(n) + m} \to \mathtt{succ(n + m)}$ (where • depicts a nat-labeled node):



□

The application of an evaluation rule $(L \hookleftarrow K \xrightarrow{b} R)$ to a jungle G requires to find an occurrence of L in G and works then analogously to the construction of K and R described above:

- Find a jungle morphism $g : L \to G$.
- Remove the edge outgoing from $g_V(root_L)$, yielding a jungle D.

[3] That is, there is a unique node $root_L$ with $indegree_L(root_L) = 0$ and only variables may have an indegree greater than 1.

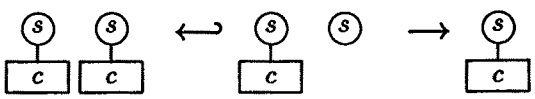
- The resulting jungle H is obtained from D like R is constructed from K: if r is a variable, then $g_V(root_L)$ is identified with $g_V(r)$; otherwise, H is the disjoint union of D and the variable-collapsed tree R' where $root_{R'}$ is identified with $g_V(root_L)$ and each $x \in VAR_{R'}$ is identified with $g_V(x)$.

The set of all evaluation rules for rewrite rules in $\mathcal{R}$ is denoted by $\mathcal{E}$. We write $G \underset{\mathcal{E}}{\Longrightarrow} H$ if H is the result of applying a rule from $\mathcal{E}$ to G and call this an *evaluation step*. For the resulting jungle H only its structure and labeling matters. So each jungle H' isomorphic[4] to H is allowed as a result of this evaluation step as well. We require, however, that variable nodes are not renamed.

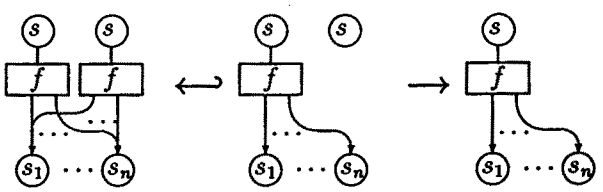
Evaluation steps rewrite certain terms represented by a jungle. But there is a problem with evaluation rules for non-left-linear rewrite rules: since the left-hand side L of such a rule is not a tree, there may be no jungle morphism from L to a jungle G although the underlying term rewrite rule is applicable to a term represented by G. This happens if a shared variable in L corresponds to different occurrences of a subterm in G. To overcome this problem we introduce *folding rules* which allow to compress a jungle such that equal subterms are represented by the same node.

For each function symbol in F there is a folding rule $(L \hookleftarrow K \xrightarrow{b} R)$:

For each constant symbol $c :\rightarrow s$:



For each function symbol $f : s_1 \dots s_n \rightarrow s$:



$\mathcal{F}$ denotes the set of all folding rules for F. The application of a folding rule to a jungle G is called a *folding step* and works as follows:

- Find a jungle morphism $g : L \rightarrow G$ with g_E injective.
- Remove $g_E(e)$, where e is the unique edge in $L - K$.
- Identify $g_V(v)$ and $g_V(v')$, where v and v' are the two source nodes in L.

If H is the resulting jungle, then this folding step is denoted by $G \underset{\mathcal{F}}{\Longrightarrow} H$. Again we allow each jungle isomorphic to H as a result of this folding step (but require that variables are not renamed).

The jungles involved in the construction of an evaluation or folding step are related by jungle morphisms in the following way:

[4]Two jungles G and H are said to be *isomorphic*, denoted by $G \cong H$, if there is a jungle morphism $g : G \rightarrow H$ with g_V and g_E bijective.

$$\begin{array}{ccccc} L & \hookleftarrow & K & \xrightarrow{b} & R \\ g\downarrow & & \downarrow & & \downarrow \\ G & \hookleftarrow & D & \xrightarrow{c} & H \end{array}$$

In fact, this diagram represents two pushouts in a category of hypergraphs and hypergraph morphisms (see [Ehr 79] for a category theoretic approach to graph rewriting).

Evaluation and folding steps are called *reduction steps* in the following and for each such reduction step we have a mapping $track : V_G \to V_H$ which coincides with c_V in the above diagram (note that $V_G = V_D$).

We conclude this section with two main theorems from [HP 88]. Let $\mathcal{G}(\mathcal{R}) = \mathcal{E} \cup \mathcal{F}$ be the set of all evaluation and folding rules for $\mathcal{R}$ and let $\underset{\mathcal{G}(\mathcal{R})}{\Longrightarrow}$ and $\overset{*}{\underset{\mathcal{G}(\mathcal{R})}{\Longrightarrow}}$ denote the reduction step relation and its transitive reflexive closure, respectively.

2.1 Normal Form Theorem
Let $G \overset{*}{\underset{\mathcal{G}(\mathcal{R})}{\Longrightarrow}} H$ be a reduction such that H is in normal form (i.e. no rule in $\mathcal{G}(\mathcal{R})$ is applicable). Then for each node v in G, $term_H(track(v))$[5] is a normal form of $term_G(v)$.

2.2 Termination Theorem
If $\mathcal{R}$ is terminating, then $\mathcal{G}(\mathcal{R})$ is terminating, too.

Both theorems together provide a completeness result for the implementation of terminating term rewrite systems by jungle evaluation. To rewrite a term t to normal form one represents t by some jungle and applies evaluation and folding rules as long as possible. This process terminates and yields a jungle which represents a normal form of t. In particular, if $\mathcal{R}$ is also confluent, then this procedure delivers the unique normal form of t.

3 Termination of Combined Reduction Systems

In this section we prove that combined jungle reduction systems inherit the termination property from their components. For $i = 0, 1$ let $\mathcal{R}_i$ be a term rewriting system with signature $SIG_i = (S_i, F_i)$ and let $\mathcal{G}_i = \mathcal{E}_i \cup \mathcal{F}_i$ be the set of all evaluation and folding rules for $\mathcal{R}_i$. All jungles considered in the following are jungles over the combined signature $(S_0 \cup S_1, F_0 \cup F_1)$.[6]

3.1 Main Theorem
Let the left-hand sides of $\mathcal{R}_i$ have no common function symbols with the right-hand sides of $\mathcal{R}_{1-i}$ $(i = 0, 1)$. Then $\mathcal{G}_0 \cup \mathcal{G}_1$ is terminating if and only if $\mathcal{G}_0$ and $\mathcal{G}_1$ are terminating. □

[5] Here *track* denotes the obvious extension of the *track*-function to sequences of reduction steps.

[6] We assume that each $f \in F_0 \cup F_1$ has unique argument and result sorts in $S_0 \cup S_1$.

Since the "only if"-direction is obvious we assume from now on that $\mathcal{G}_0$ and $\mathcal{G}_1$ are terminating (and that $\mathcal{R}_0$ and $\mathcal{R}_1$ satisfy the disjointness condition of the theorem).

For $i = 0, 1$ let $SIG_i^{\ominus}$ be the subsignature of SIG_i consisting of all function symbols occurring in the rules of $\mathcal{R}_i$ and all associated argument and result sorts; $LEFT(\mathcal{R}_i)$ denotes the set of all function symbols occurring in the left-hand sides of $\mathcal{R}_i$. For each jungle G and $i = 0, 1$ let

- G^i be the subjungle[7] consisting of all nodes and edges with label in $SIG_i^{\ominus}$,
- $\mathcal{L}(G^i)$ be the subjungle consisting of all edges with label in $LEFT(\mathcal{R}_i)$ and all source and target nodes of these edges (hence $\mathcal{L}(G^i)$ is a subjungle of G^i), and
- $G^{\#}$ be the number of edges with label in $LEFT(\mathcal{R}_0) \cap LEFT(\mathcal{R}_1)$.

For proving theorem 3.1 we exploit the fact that in each reduction over $\mathcal{G}_0 \cup \mathcal{G}_1$ the number $G^{\#}$ remains constant after a finite number of steps (lemma 3.2). From this point on the application of $\mathcal{G}_i$-rules affects G^{1-i} in a way described by an extension of $\underset{\mathcal{G}_{1-i}}{\Longrightarrow}$ which is still terminating. As a consequence, every infinite reduction over $\mathcal{G}_0 \cup \mathcal{G}_1$ can be transformed into an infinite sequence of extended $\underset{\mathcal{G}_i}{\Longrightarrow}$-steps for some $i \in \{0, 1\}$.

3.2 Lemma
For all jungles G, H:

$$G \underset{\mathcal{G}_0 \cup \mathcal{G}_1}{\Longrightarrow} H \text{ implies } G^{\#} \geq H^{\#}.$$

Proof
Let $G \underset{\mathcal{G}_0 \cup \mathcal{G}_1}{\Longrightarrow} H$ through a rule $r = (L \hookleftarrow K \overset{b}{\rightarrow} R)$. Then $H^{\#} = (G^{\#} - L^{\#}) + R^{\#}$. So it suffices to show $L^{\#} \geq R^{\#}$. Folding rules have this property by construction. If r is an evaluation rule, then $L^{\#} \geq K^{\#} = R^{\#}$ since, by assumption, no symbol in $LEFT(\mathcal{R}_0) \cap LEFT(\mathcal{R}_1)$ occurs in any right-hand side of $\mathcal{R}_0 \cup \mathcal{R}_1$. □

The key to the proof of theorem 3.1 is to consider the "fusion" of two jungle nodes, where at least one of the nodes is a variable. We define a relation $\rightsquigarrow$ on jungles as follows:

$$G \rightsquigarrow H \quad \text{if} \quad \begin{array}{l} H \text{ is (isomorphic to a jungle) obtained from } G \text{ by} \\ \text{identification of a variable with some other node.} \end{array}$$

3.3 Lemma
For $i = 0, 1$, the relation $\underset{\mathcal{G}_i}{\Longrightarrow} \cup \rightsquigarrow$ is terminating on SIG_i-jungles.

Proof
Each $\rightsquigarrow$-step decreases the number of variables in a jungle by one while $\underset{\mathcal{G}_i}{\Longrightarrow}$-steps do not change this number. Hence each sequence of $\underset{\mathcal{G}_i}{\Longrightarrow}$- and $\rightsquigarrow$-steps contains only a finite number of $\rightsquigarrow$-steps and the proposition follows from the termination of $\underset{\mathcal{G}_i}{\Longrightarrow}$. □

[7] A jungle U is called a *subjungle* of a jungle G, denoted by $U \subseteq G$, if $V_U \subseteq V_G$, $E_U \subseteq E_G$, and if s_U, t_U, l_U, m_U are restrictions of the corresponding mappings of G.

In the following $\mathcal{F}$ stands for the set $\mathcal{F}_0 \cup \mathcal{F}_1$ of all folding rules. It is clear that $\Longrightarrow_{\mathcal{F}}$ is terminating since each folding step decreases the number of nodes by one.

The next lemma reveals the basic idea for the proof of theorem 3.1. It describes the effect of an $\mathcal{G}_i$-step on the SIG_{1-i}-part of a jungle.

3.4 Lemma
Let $G \Longrightarrow_{\mathcal{E}_i \cup \mathcal{F}} H$ and $U \subseteq G^{1-i}$ for some $i \in \{0,1\}$. If $G^{\#} = H^{\#}$, then there is $\tilde{U} \subseteq H^{1-i}$ such that

$$U \cong \tilde{U} \text{ or } U \rightsquigarrow \tilde{U} \text{ or } U \Longrightarrow_{\mathcal{F}_{1-i}} \tilde{U}.$$

Proof
Case 1: $G \Longrightarrow_{\mathcal{E}_i} H$. Let e be the edge in $\mathcal{L}(G^i)$ which is deleted. By assumption, the evaluation step does not create edges with label in $LEFT(\mathcal{R}_{1-i})$. With $G^{\#} = H^{\#}$ follows $e \notin \mathcal{L}(G^{1-i})$, as otherwise $G^{\#} > H^{\#}$.

We conclude $e \notin G^{1-i}$ since the label of e does not occur in any right-hand side of $\mathcal{R}_{1-i}$. So the evaluation step can change the structure of U only through the identification of two nodes. (That is, if no such identification takes place, then H^{1-i} contains an isomorphic copy $\tilde{U}$ of U.) Assume that there are nodes v_1, v_2 in U with $v_1 \neq v_2$ but $track(v_1) = track(v_2)$. Then either v_1 or v_2 is the source node of e. But this node is a variable in U because $e \notin G^{1-i}$. Hence $U \rightsquigarrow \tilde{U}$ with $\tilde{U} \subseteq H^{1-i}$.

Case 2: $G \Longrightarrow_{\mathcal{F}} H$. There is a unique surjective jungle morphism $fold : G \to H$ with $fold_V = track$. Let e_1, e_2 be the two edges in G with $e_1 \neq e_2$ but $fold(e_1) = fold(e_2)$. If $e_1, e_2 \in U$, then the applied folding rule is in $\mathcal{F}_{1-i}$ and we have $U \Longrightarrow_{\mathcal{F}_{1-i}} fold(U)$ where $fold(U) \subseteq H^{1-i}$. Otherwise, a simple case analysis shows that either $U \cong fold(U)$ or $U \rightsquigarrow fold(U)$. □

To assemble the proof of the main theorem we need one more lemma which allows to restrict evaluation steps to subjungles.

3.5 Lemma
Let $G \Longrightarrow_{\mathcal{E}_i} H$ and $U \subseteq G^i$ for some $i \in \{0,1\}$. If $\mathcal{L}(G^i) \subseteq U$, then $U \Longrightarrow_{\mathcal{E}_i} X$ where $\mathcal{L}(H^i) \subseteq X \subseteq H^i$.

Proof
If $\mathcal{L}(G^i) \subseteq U$, then the occurrence of the left-hand side of the applied rule lies in U. Hence there is a restricted step $U \Longrightarrow_{\mathcal{E}_i} X \subseteq H$ (for a proof see the CLIP-theorem in [Kre 77]). Clearly we have $X \subseteq H^i$. Moreover, $\mathcal{L}(H^i) \subseteq X$ holds since all edges produced by the evaluation step belong to X. □

3.6 Proof of the Main Theorem
Suppose that there is an infinite sequence

$$G_1 \Longrightarrow_{\mathcal{G}_0 \cup \mathcal{G}_1} G_2 \Longrightarrow_{\mathcal{G}_0 \cup \mathcal{G}_1} G_3 \Longrightarrow_{\mathcal{G}_0 \cup \mathcal{G}_1} \cdots$$

The sequence contains infinitely many $\Longrightarrow_{\mathcal{E}_i}$-steps for some $i \in \{0,1\}$, since $\Longrightarrow_{\mathcal{F}}$ is terminating. Without loss of generality we may assume that there are infinitely many

$\underset{\mathcal{E}_0}{\Longrightarrow}$-steps. Lemma 3.2 implies that $G_n^\#$ remains constant after a finite number of steps. Hence there is an infinite sequence

$$H_1 \underset{\mathcal{E}_0}{\Longrightarrow} H_2 \overset{*}{\underset{\mathcal{E}_1 \cup \mathcal{F}}{\Longrightarrow}} H_3 \underset{\mathcal{E}_0}{\Longrightarrow} H_4 \overset{*}{\underset{\mathcal{E}_1 \cup \mathcal{F}}{\Longrightarrow}} \dots$$

such that $H_n^\# = H_{n+1}^\#$ for $n \geq 1$.

Consider some step $H \underset{\mathcal{E}_1 \cup \mathcal{F}}{\Longrightarrow} \overline{H}$ in the above sequence and some subjungle U of H satisfying $\mathcal{L}(H^0) \subseteq U \subseteq H^0$. By lemma 3.4 there is $\tilde{U} \subseteq \overline{H}^0$ such that $U \cong \tilde{U}$ or $U \rightsquigarrow \tilde{U}$ or $U \underset{\mathcal{F}_0}{\Longrightarrow} \tilde{U}$. We also have $\mathcal{L}(\overline{H}^0) \subseteq \tilde{U}$ because the reduction step does not create edges with label in $LEFT(\mathcal{R}_0)$. Combined with lemma 3.5, this shows that there is an infinite sequence

$$U_1 \underset{\mathcal{E}_0 \cup \mathcal{F}_0}{\Longrightarrow} U_2 \overset{*}{\rightsquigarrow} U_3 \underset{\mathcal{E}_0 \cup \mathcal{F}_0}{\Longrightarrow} U_4 \overset{*}{\rightsquigarrow} \dots$$

where $U_1 = H_1^0$ and where $\overset{*}{\rightsquigarrow}$ denotes the transitive isomorphic closure of $\rightsquigarrow$. But this contradicts the termination of $\underset{\mathcal{E}_0 \cup \mathcal{F}_0}{\Longrightarrow} \cup \rightsquigarrow$ (lemma 3.3) since all the U_n are SIG_0-jungles. □

References

[BCV 85] M. Bidoit, C. Choppy, F. Voisin: *The ASSPEGIQUE Specification Environment: Motivations and Design.* In H.-J. Kreowski (ed.): *Recent Trends in Data Type Specification.* Informatik-Fachberichte 116, Springer-Verlag (1985)

[BHK 89] J. Bergstra, J. Heering, P. Klint: *Algebraic Specification.* ACM Press (1989)

[BD 86] L. Bachmair, N. Dershowitz: *Commutation, Transformation, and Termination.* Proc. 8th CADE, Lecture Notes in Comp. Sci. 230, 5-20 (1986)

[DJ 90] N. Dershowitz, J.-P. Jouannaud: *Rewrite Systems.* In J. van Leeuwen (ed.): *Handbook of Theoretical Computer Science.* Vol. B, North-Holland (1990)

[Dro 89] K. Drosten: *Termersetzungssysteme.* Informatik-Fachberichte 210, Springer-Verlag (1989) (In German)

[Ehr 79] H. Ehrig: *Introduction to the Algebraic Theory of Graph Grammars.* Proc. 1st Graph Grammar Workshop, Lecture Notes in Comp. Sci. 73, 1-69 (1979)

[EM 85] H. Ehrig, B. Mahr: *Fundamentals of Algebraic Specification 1 – Equations and Initial Semantics.* Springer-Verlag (1985)

[FGJM 85] K. Futatsugi, J. Goguen, J.P. Jouannoud, J. Meseguer: *Principles of OBJ2.* Proc. 1985 Symposium on Principles of Programming Languages, 52-66 (1985)

[GG 87] H. Ganzinger, R. Giegerich: *A Note on Termination in Combinations of Heterogeneous Term Rewriting Systems.* EATCS Bulletin 31, 22-28 (1987)

[GH 86] A. Geser, H. Hußmann: *Experiences with the RAP System: A Specification Interpreter Combining Term Rewriting and Resolution.* Proc. ESOP '86, Lecture Notes in Comp. Sci. 213, 339-350 (1986)

[HKP 88] A. Habel, H.-J. Kreowski, D. Plump: *Jungle Evaluation.* Proc. Fifth Workshop on Specification of Abstract Data Types, Lecture Notes in Comp. Sci. 332, 92-112 (1988)

[HP 88] B. Hoffmann, D. Plump: *Jungle Evaluation for Efficient Term Rewriting.* Proc. Algebraic and Logic Programming, Akademie-Verlag, Berlin (GDR), 191-203 (1988). Also in Lecture Notes in Comp. Sci. 343, (1989). Long version available as technical report no. 4/88, Universität Bremen

[Klo 90] J.W. Klop: *Term Rewriting Systems – From Church-Rosser to Knuth-Bendix and Beyond.* Proc. ICALP '90, Lecture Notes in Comp. Sci. 443, 350-369 (1990)

[Kre 77] H.-J. Kreowski: *Manipulationen von Graphmanipulationen.* Dissertation, TU Berlin (1977)

[Mid 89a] A. Middeldorp: *A Sufficient Condition for the Termination of the Direct Sum of Term Rewriting Systems.* Proc. 4th IEEE Symposium on Logic in Computer Science, 396-401 (1989)

[Mid 89b] A. Middeldorp: *Termination of Disjoint Unions of Conditional Term Rewriting Systems.* Report CS-R8959, Centre for Mathematics and Computer Science, Amsterdam (1989)

[Rus 87] M. Rusinowitch: *On Termination of the Direct Sum of Term Rewriting Systems.* Information Process. Lett. 26, 65-70 (1987)

[TKB 89] Y. Toyama, J.W. Klop, H.P. Barendregt: *Termination for the Direct Sum of Left-Linear Term Rewriting Systems.* Proc. Rewriting Techniques and Applications '89, Lecture Notes in Comp. Sci. 355, 477-491 (1989)

[Toy 87a] Y. Toyama: *On the Church-Rosser Property for the Direct Sum of Term Rewriting Systems.* Journal of the ACM 34, 128-143 (1987)

[Toy 87b] Y. Toyama: *Counterexamples to Termination for the Direct Sum of Term Rewriting Systems.* Information Process. Lett. 25, 141-143 (1987)