On Termination of Graph Rewriting

Detlef Plump*

Universität Bremen, Fachbereich 3 — Informatik
Postfach 33 04 40, D-28334 Bremen, Germany
e-mail: det@informatik.uni-bremen.de

Abstract

A necessary and sufficient condition for termination of graph rewriting systems is established. Termination is equivalent to the finiteness of all *forward closures*, being certain minimal derivations in which each step depends on previous steps. This characterization differs from corresponding results for term rewriting in that the latter hold only for subclasses of term rewriting systems. When applied to term graph rewriting, the result characterizes termination of arbitrary term rewriting systems under graph rewriting. In particular, it captures non-terminating term rewriting systems that are terminating under graph rewriting.

1 Introduction

Proving that a program terminates for all inputs is an important but in general difficult task (in fact, the problem is unsolvable in general). In the area of term rewriting systems, proof methods for termination have been investigated for a long time, leading to a comprehensive theory of termination (see Dershowitz's survey [Der87]). For graph rewriting systems, however, termination criteria have not yet been developed to the author's knowledge. (Robertson and Seymour [RS85] have proved a fundamental well-ordering result for graphs, but it seems that this has not been applied to graph rewriting up to now.)

The objective of this paper is to show that termination of graph rewriting is equivalent to termination of certain restricted derivations. These so-called forward closures are derivations in which each step depends on previous steps, and that are minimal in the sense that all nodes and edges are used or generated by the rules. Forward closures can be inductively generated from the rules, providing a proof method for termination which attempts to show that only finite forward closures are possible.

For term rewriting systems, Dershowitz [Der81] and Geupel [Geu89] have shown that right-linear respectively non-overlapping rewrite rules terminate if their forward closures do. But, different from the present result, there are counterexamples demonstrating that terminating forward closures need not guaran-

*Research partially supported by ESPRIT Basic Research Working Group 6112, *COMPASS*

tee termination in general. In contrast, the present result characterizes termination of arbitrary term rewriting systems in the computational model *term graph rewriting*. In particular, the result enables termination proofs for non-terminating term rewriting systems that are terminating under graph rewriting (see Section 4).

2 Graph rewriting

In this section a hypergraph variant of the algebraic approach to graph rewriting is briefly reviewed. For more information, the reader is referred to Ehrig's introduction [Ehr79] and Habel's recent survey [Hab92].

2.1 Hypergraphs and hypergraph morphisms

Let Σ be a *signature*, that is, a set such that each $l \in \Sigma$ comes with a natural number $type(l) \geq 0$. A *hypergraph* over Σ is a system $G = \langle V_G, E_G, lab_G, att_G \rangle$ consisting of two finite sets V_G and E_G of *nodes* and *hyperedges*, a labelling function $lab_G \colon E_G \to \Sigma$, and an attachment function $att_G \colon E_G \to V_G^*$ which assigns a sequence of nodes to each hyperedge e such that $type(lab_G(e))$ is the length of $att_G(e)$.

In pictures, a hyperedge e is represented as a box labelled with $lab_G(e)$. The box is connected with the attachment nodes in $att_G(e)$ by lines. For example, Figure 1 shows a hypergraph with three hyperedges, where $type(\texttt{f}) = 4$ and $type(\texttt{a}) = type(\texttt{b}) = 1$ (for simplicity, the order among the attachment nodes of the `f`-hyperedge is not indicated).

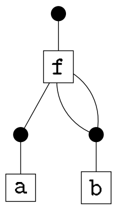


Figure 1: A hypergraph

Let G,H be hypergraphs. Then G is a *subhypergraph* of H, denoted by $G \subseteq H$, if $V_G \subseteq V_H$, $E_G \subseteq E_H$, and if lab_G and att_G are restrictions of lab_H and att_H.

A *hypergraph morphism* $f \colon G \to H$ consists of two functions $f_V \colon V_G \to V_H$ and $f_E \colon E_G \to E_H$ that preserve labels and attachment to nodes, that is, $lab_H \circ f_V = lab_G$ and $att_H \circ f_E = f_V^* \circ att_G$. (The extension $f^* \colon A^* \to B^*$ of a function $f \colon A \to B$ maps the empty string to itself and $a_1 \ldots a_n$ to $f(a_1) \ldots f(a_n)$.) The morphism f is *injective* (*surjective*) if f_V and f_E are injective (surjective). A morphism that is both injective and surjective is an *isomorphism*.

2.2 Rules and derivations

A *rule* $r = (L \leftarrow K \rightarrow R)$ consists of two hypergraph morphisms, where $K \rightarrow L$ is injective. L and R are called the *left-hand side* and the *right-hand side* of r.

Given two hypergraphs G and H, a *direct derivation* from G to H by r consists of two hypergraph pushouts[1] of the form shown in Figure 2. Intuitively,

$$\begin{tikzcd} L \arrow[d, "g"'] & K \arrow[l] \arrow[r] \arrow[d] & R \arrow[d] \\ G & D \arrow[l] \arrow[r] & H \end{tikzcd}$$

Figure 2: A direct derivation

D is obtained from G by removing the nodes and hyperedges in $g(L - K')$, where K' is the image of K in L. H is constructed from D by identifying items in D according to the morphism $K \rightarrow R$ and by adding all items of R that are not in the image of K. Such a direct derivation is denoted by $G \Rightarrow_{r,g} H$ or $G \Rightarrow_r H$ or simply by $G \Rightarrow H$. The subhypergraph $g(L)$ of G is the *redex* of this direct derivation and is denoted by $Redex(G \Rightarrow H)$.

A *derivation* from G to H, denoted by $G \Rightarrow^* H$, is either an isomorphism $G \rightarrow H$ (which is a derivation of length 0) or a non-empty sequence of direct derivations of the form $G = G_0 \Rightarrow G_1 \Rightarrow \ldots \Rightarrow G_n = H$. A derivation of the latter kind is also denoted by $G \Rightarrow^+ H$. An *infinite derivation* is an infinite sequence of the form $G_0 \Rightarrow G_1 \Rightarrow G_2 \Rightarrow \ldots$

A *graph rewriting system* $\mathcal{G} = (\Sigma, \mathcal{R})$ consists of a signature Σ and a set $\mathcal{R}$ of rules with hypergraphs over Σ.

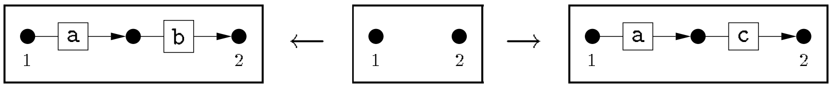
3 Termination

In the first part of this section, the concept of independent derivation steps is recalled and generalized. Based on this concept, in the second part forward closures are introduced as special derivations and the main result of this paper is stated. The third part of this section is devoted to the proof of the main theorem.

Definition 3.1 (termination) A graph rewriting system $\mathcal{G}$ is *terminating* if it does not admit an infinite derivation.

[1] See [Ehr79] for the definition and construction of graph pushouts. The generalization to the hypergraph case is straightforward.

Example 3.2 Suppose that $\mathcal{G}$ contains only the following rule[2]:



$\mathcal{G}$ is terminating since every rule application reduces the number of `b`-labels in a hypergraph by one.

3.1 Independence of derivations

Definition 3.3 (independence) Two direct derivations

$$\begin{tikzcd}
L \arrow[d] & K \arrow[l] \arrow[r] \arrow[d] & R \arrow[dr, "h"'] & & L' \arrow[dl, "h'"] & K' \arrow[l] \arrow[r] \arrow[d] & R' \arrow[d] \\
G & D \arrow[l] \arrow[rr, "c"] & & H & & D' \arrow[ll, "c'"'] \arrow[r] & M
\end{tikzcd}$$

are *independent* if there are hypergraph morphisms $f\colon R \to D'$ and $f'\colon L' \to D$ such that $c' \circ f = h$ and $c \circ f' = h'$.

A necessary condition for independence is that $G \Rightarrow H$ does not generate items that are used by $H \Rightarrow M$ and that $H \Rightarrow M$ does not remove items that are used by $G \Rightarrow H$. More precisely, $h(R) \cap h'(L') \subseteq h(b(K)) \cap h'(b'(K'))$ must hold, where b and b' are the morphisms $K \to R$ and $K' \to L'$. This condition is sufficient for independence if the two rules are non-identifying, that is, if $K \to R$ and $K' \to R'$ are injective.

Independence is an important property as it allows to interchange rule applications. The following theorem was established by Ehrig and Kreowski for the graph case. Its proof can be easily adapted to the present setting.

Theorem 3.4 ([EK76]) *Let $G \Rightarrow_q H \Rightarrow_r M$ be independent direct derivations. Then there are direct derivations $G \Rightarrow_r H' \Rightarrow_q M$.*

The steps $G \Rightarrow_r H' \Rightarrow_q M$ need not be unique since identifying rules are allowed. But for the following considerations only the existence of such steps matters, not their uniqueness.

Definition 3.5 (generalized independence) A derivation $G \Rightarrow^* M$ and a direct derivation $M \Rightarrow_r N$ are *independent* if either

(1) $G \Rightarrow^* M$ has length 0, or

(2) $(G \Rightarrow^* M) = (G \Rightarrow^* H \Rightarrow_q M)$ and

[2] Node numbers indicate the hypergraph morphisms, and arrows are used to distinguish the second from the first attachment node of a hyperedge.

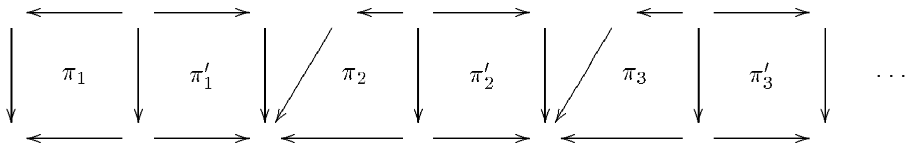
- $H \Rightarrow_q M \Rightarrow_r N$ are independent (in the sense of Definition 3.3),
- $G \Rightarrow^* H$ and $H \Rightarrow_r M'$ are independent for some $H \Rightarrow_r M'$ such that $H \Rightarrow_r M' \Rightarrow_q N$ exist by Theorem 3.4.

Independence of $G \Rightarrow^* M$ and $M \Rightarrow_r N$ means that the step $M \Rightarrow_r N$ can be shifted to the beginning, yielding a derivation $G \Rightarrow_r G' \Rightarrow^* N$. Two derivations that are not independent are said to be *dependent*.

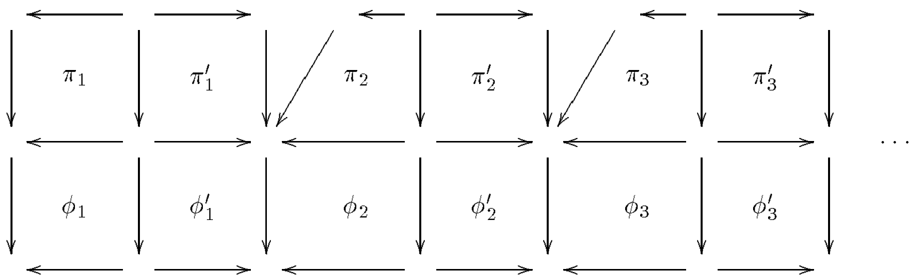
3.2 Forward closures and termination

Forward closures are derivations in which each step depends on previous steps, and that are minimal in the sense that all nodes and hyperedges occur in some redex or are generated by the rules. To make this precise, instances of derivations are considered. An instance of a (possibly infinite) derivation is obtained by extending it with "context". Formally, an instance is a derivation whose pushouts are composed from the given pushouts and suitable new pushouts.

Definition 3.6 (instance) Consider a finite or infinite derivation[3]



A derivation is an *instance* (or *embedding*) of this derivation if it can be written



where for each $i \geq 1$, ϕ_i and ϕ'_i are pushouts with injective vertical morphisms. (So the pushouts constituting the derivation are composed of the pushouts π_i and ϕ_i, respectively π'_i and ϕ'_i.)

Given an instance $G \Rightarrow^+ H$ of a derivation $G^\ominus \Rightarrow^+ H^\ominus$ with associated injective morphism $f\colon H^\ominus \to H$, the difference $H - f(H^\ominus)$ (consisting of two sets of nodes and hyperedges) is denoted by $New(H)$. By pushout properties, hyperedges in $New(H)$ can have attachment nodes in $H^\ominus$ only if these nodes exist already in $G^\ominus$ and are not removed by any step in $G^\ominus \Rightarrow^+ H^\ominus$.

[3] The hypergraphs in the pushout diagrams are omitted to simplify the presentation.

Definition 3.7 (forward closure) *Forward closures* are inductively defined as follows:

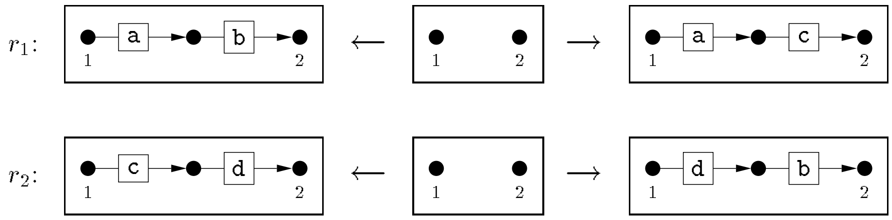
(1) Every direct derivation $L \Rightarrow_{r,g} R$ with surjective g is a forward closure.

(2) A derivation $G \Rightarrow^+ H \Rightarrow M$ is a forward closure if $G \Rightarrow^+ H$ is an instance of a forward closure, and $H \Rightarrow M$ is a direct derivation such that

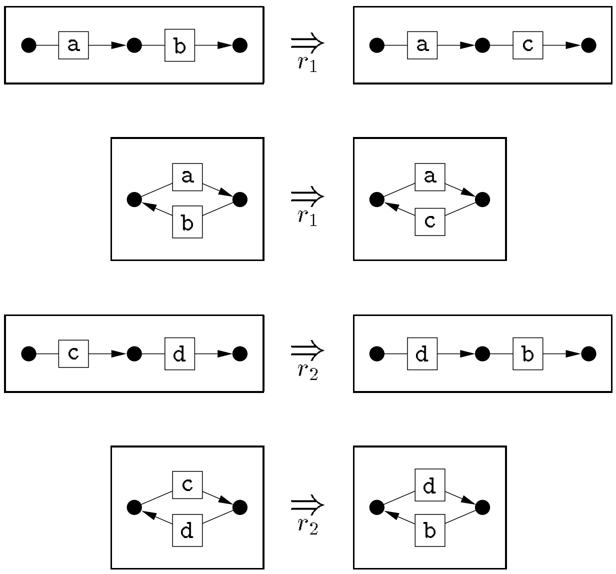
- $G \Rightarrow^+ H$ and $H \Rightarrow M$ are dependent,
- $New(H) \subseteq Redex(H \Rightarrow M)$.

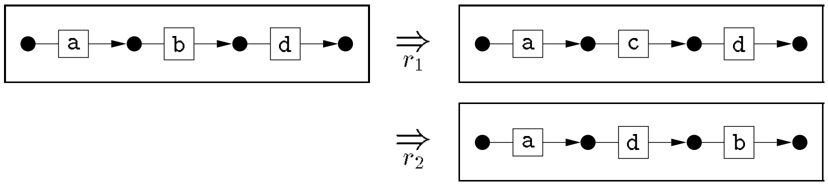
Note that the prefix $G \Rightarrow^+ H$ of $G \Rightarrow^+ H \Rightarrow M$ is not a forward closure in general. This happens only in the special case where $G \Rightarrow^+ H$ has been taken as an instance of itself.

Example 3.8 Consider the following two rules:



This system has (up to isomorphism) five forward closures:



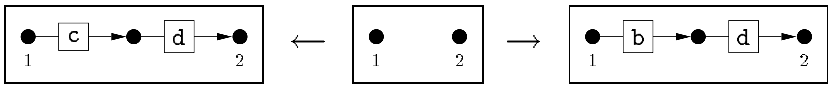


The last forward closure is constructed by applying r_2 to an instance of the first forward closure.

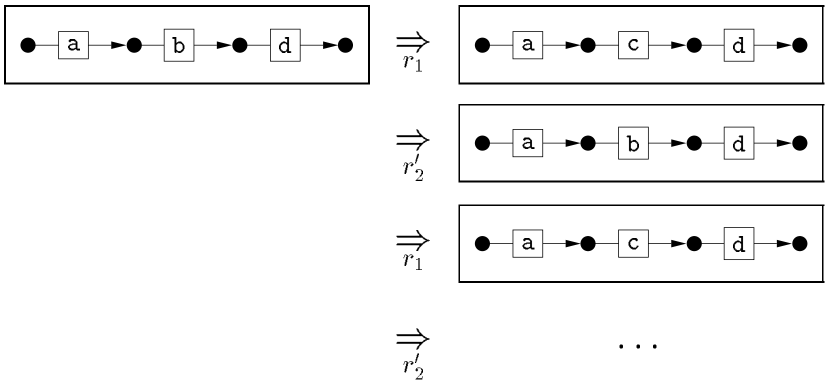
Definition 3.9 (infinite forward closure) An *infinite forward closure* is an infinite derivation $G_0 \Rightarrow G_1 \Rightarrow G_2 \Rightarrow \ldots$ that contains a forward closure as a prefix. That is, there is some $n \geq 1$ such that $G_0 \Rightarrow^+ G_n$ is a forward closure.

If each rule removes at least one item (and only such systems will matter in the following, see below), then every extension $G_0 \Rightarrow^+ G_{n+i}$ of the prefix $G_0 \Rightarrow^+ G_n$ is also a forward closure. This is because for every $p \geq n$, the step $G_p \Rightarrow G_{p+1}$ removes an item in the image of the right-hand side of some rule in $G_0 \Rightarrow^+ G_p$.

Example 3.10 Suppose that the rule r_2 of Example 3.8 is modified by exchanging the labels `d` and `b` in the right-hand side, yielding the following rule:

r_2': 

Now the modified system admits an infinite forward closure:



It is important to notice that a graph rewriting system is trivially non-terminating if it contains a rule that does not remove anything. This is because such a rule can be applied to its own right-hand side. Thus, the absence of non-removing rules is a necessary condition for termination.

Definition 3.11 A graph rewriting system is *consumptive* if each rule $(L \leftarrow K \rightarrow R)$ has a non-surjective morphism $K \rightarrow L$.

Theorem 3.12 (main theorem) *A graph rewriting system is terminating if, and only if, it is consumptive and does not admit an infinite forward closure.*

Example 3.13 The system in Example 3.8 is terminating as it is consumptive and possesses only finite forward closures. Note that for this system, termination cannot be proved by comparing the sizes of hypergraphs or the numbers of labels before and after rule applications. Both rules are size-preserving, and after two consecutive steps with r_1 and r_2, the numbers of `a`'s and `b`'s in a hypergraph is the same as before.

3.3 Proof of the main theorem

This subsection contains the lemmas and additional definitions needed for the proof of Theorem 3.12. The essence of the proof is provided by Lemmas 3.17 and 3.19. The former shows that every infinite derivation in which each step depends on previous steps can be transformed into an infinite forward closure. Lemma 3.19 shows that every infinite derivation of a consumptive system can be reordered such that each step depends on previous steps.

Definition 3.14 (dependence chain) A *dependence chain* is a derivation $G_0 \Rightarrow \ldots \Rightarrow G_n$ such that $G_0 \Rightarrow^+ G_i$ and $G_i \Rightarrow G_{i+1}$ are dependent for $i = 1, \ldots, n-1$. An *infinite dependence chain* is an infinite derivation $G_0 \Rightarrow G_1 \Rightarrow \ldots$ such that $G_0 \Rightarrow^+ G_n$ is a dependence chain for each $n \geq 1$.

The idea is to transform an infinite dependence chain into an infinite forward closure by cutting off all items that neither occur in any redex nor are generated by the rules. To formalize this, a partial hypergraph morphism *track* is introduced which maps the initial hypergraph of a derivation into the final one.

Definition 3.15 (track) Given a direct derivation $G \Rightarrow H$, the partial hypergraph morphism $track_{G \Rightarrow H}: G \rightarrow H$ is defined by

$$track_{G \Rightarrow H}(x) = \begin{cases} c'(c^{-1}(x)) & \text{if } x \in c(D), \\ \text{undefined} & \text{otherwise,} \end{cases}$$

where $c: D \rightarrow G$ and $c': D \rightarrow H$ are the morphisms in the lower row of Figure 2. (The morphism c is injective by pushout properties and hence there is an inverse morphism $c^{-1}: c(D) \rightarrow D$.) For a derivation $G = G_0 \Rightarrow G_1 \Rightarrow \ldots \Rightarrow G_n = H$, $track_{G \Rightarrow^+ H}$ is defined by

$$track_{G \Rightarrow^+ H} = track_{G_{n-1} \Rightarrow G_n} \circ \ldots \circ track_{G_0 \Rightarrow G_1}.$$

The above described "clipping" of a derivation amounts to remove from the initial hypergraph all items that will never occur in a redex, and to remove from the subsequent hypergraphs all descendants of these items.

Lemma 3.16 (clipping lemma) *Let* $\Delta\colon G_0 \Rightarrow G_1 \Rightarrow \ldots$ *be a finite or infinite derivation and* C_0 *be the subhypergraph of* G_0 *consisting of* $\mathit{Redex}(G_0 \Rightarrow G_1)$ *and all items* x *such that* $\mathit{track}_{G_0 \Rightarrow^+ G_i}(x) \in \mathit{Redex}(G_i \Rightarrow G_{i+1})$ *for some* $i \geq 1$. *Then there is a derivation* $\mathit{Clip}(\Delta)\colon C_0 \Rightarrow C_1 \Rightarrow \ldots$ *such that* Δ *is an instance of* $\mathit{Clip}(\Delta)$.

Proof Straightforward generalization of the corresponding proof for finite derivations in [Kre77]. □

Lemma 3.17 *If a graph rewriting system admits an infinite dependence chain, then it admits an infinite forward closure.*

Proof Let $\Delta\colon G_0 \Rightarrow G_1 \Rightarrow \ldots$ be an infinite dependence chain. Then $\mathit{Clip}(\Delta)$: $C_0 \Rightarrow C_1 \Rightarrow \ldots$ is also a dependence chain since clipping preserves dependence and independence of each two subsequent steps in Δ.

Claim. For each $i \geq 1$, $\mathit{Clip}(C_0 \Rightarrow^+ C_i)$ is a forward closure.

Proof. By definition of clipping, $\mathit{Clip}(C_0 \Rightarrow C_1)$ is a step $L \Rightarrow R$ with $L = \mathit{Redex}(C_0 \Rightarrow C_1)$. Clearly, this is a forward closure. Now suppose, as induction hypothesis, that $C_0 \Rightarrow^+ C_n$ satisfies the claim for some $n \geq 1$. Let $\mathit{Clip}(C_0 \Rightarrow^+ C_n \Rightarrow C_{n+1}) = (B_0 \Rightarrow^+ B_n \Rightarrow B_{n+1})$. Then $B_0 \Rightarrow^+ B_n$ is an instance of $\mathit{Clip}(C_0 \Rightarrow^+ C_n)$, and $\mathit{New}(B_n) = \mathit{track}_{C_0 \Rightarrow^+ C_n}(X)$ where X consists of all x whose descendants occur in no redex of $C_0 \Rightarrow^+ C_n$ but in $\mathit{Redex}(C_n \Rightarrow C_{n+1})$. Since $\mathit{Redex}(C_n \Rightarrow C_{n+1}) = \mathit{Redex}(B_n \Rightarrow B_{n+1})$, it follows by induction hypothesis that $B_0 \Rightarrow^+ B_n \Rightarrow B_{n+1}$ is a forward closure ($B_0 \Rightarrow^+ B_n$ and $B_n \Rightarrow B_{n+1}$ are dependent because $C_0 \Rightarrow^+ C_n$ and $C_n \Rightarrow C_{n+1}$ are so). □

By the claim, it suffices to show that there is some k with $\mathit{Clip}(C_0 \Rightarrow^+ C_k) = (C_0 \Rightarrow^+ C_k)$. Since $C_0 \Rightarrow C_1 \Rightarrow \ldots$ results from clipping, each item in C_0 occurs in some redex. Hence there is some $k \geq 1$ such that each item occurs in a redex of $C_0 \Rightarrow^+ C_k$. But then, obviously, $\mathit{Clip}(C_0 \Rightarrow^+ C_k)$ is $C_0 \Rightarrow^+ C_k$. □

By the virtue of Lemma 3.17, proving that a non-terminating consumptive graph rewriting system has an infinite forward closure reduces to proving that there is an infinite dependence chain. The latter is achieved by using a transformation process on infinite derivations which involves the shifting of rule applications. Given a rule application that is independent from all previous steps in a derivation, the *Shift* operation defined next moves this application to the beginning of the derivation.

Definition 3.18 (shift)

(1) $\mathit{Shift}(G \Rightarrow^* M \Rightarrow_r N) = (G \Rightarrow_r N)$ if $G \Rightarrow^* M$ has length 0.

(2) $\mathit{Shift}(G \Rightarrow^* H \Rightarrow_q M \Rightarrow_r N) = (G \Rightarrow^* M' \Rightarrow_q N)$ if $G \Rightarrow^* H \Rightarrow_q M$ and $M \Rightarrow_r N$ are independent and $(G \Rightarrow^* M') = \mathit{Shift}(G \Rightarrow^* H \Rightarrow_r M')$, where $H \Rightarrow_r M' \Rightarrow_q N$ result from interchanging $H \Rightarrow_q M \Rightarrow_r N$ such that $G \Rightarrow^* H \Rightarrow_r M'$ are independent.

Lemma 3.19 *Every non-terminating consumptive graph rewriting system admits an infinite dependence chain.*

Proof Let $G_0 \Rightarrow G_1 \Rightarrow \ldots$ be an infinite derivation. There is some $n \geq 1$ such that $track_{G_0 \Rightarrow^+ G_j}(G_0)$ has the same size for each $j \geq n$ (since the size of $track_{G_0 \Rightarrow^+ G_i}(G_0)$ does not increase for growing i). As the given system is consumptive, this implies the following:

For each $j \geq n$, $G_j \Rightarrow G_{j+1}$ removes an item generated by $G_0 \Rightarrow^+ G_j$. $\quad (*)$

(An item in G_j is *generated* by $G_0 \Rightarrow^+ G_j$ if it is not in $track_{G_0 \Rightarrow^+ G_j}(G_0)$.) Now consider the n steps $G_0 \Rightarrow \ldots \Rightarrow G_n$. Each of these steps is (trivially) a dependence chain. An infinite transformation process is used to show that at least one of these chains can be extended to an infinite dependence chain. Each step of the process transforms a derivation $G_0 \Rightarrow^+ G_j$ composed from n dependence chains into a derivation $G_0 \Rightarrow^+ G_{j+1}$ that again consists of n dependence chains. Thus, at least one of the chains must grow ad infinitum. Each step of the process is realized by applying two transformation rules. In the following description of these rules, derivations on top of the bar are transformed into those below the bar, and framed derivations represent dependence chains.

extend: $\dfrac{\boxed{X \Rightarrow^+ Y} \Rightarrow_r Z}{\boxed{X \Rightarrow^+ Z}}$ where $X \Rightarrow^+ Y$ and $Y \Rightarrow_r Z$ are dependent

shift: $\dfrac{\boxed{X \Rightarrow^+ Y} \Rightarrow_r Z}{X \Rightarrow_r \boxed{X' \Rightarrow^+ Z}}$ where $X \Rightarrow^+ Y$ and $Y \Rightarrow_r Z$ are independent and $Shift(X \Rightarrow^+ Y \Rightarrow_r Z) = (X \Rightarrow_r X' \Rightarrow^+ Z)$

It remains to show that each derivation $G_0 \Rightarrow^+ G_j \Rightarrow_r G_{j+1}$, where $G_0 \Rightarrow^+ G_j$ consists of n dependence chains, can be transformed into a derivation $G_0 \Rightarrow^+ G_{j+1}$ consisting of n dependence chains. Starting with the n^{th} dependence chain, repeated application of **shift** leads to a segment $X \Rightarrow^+ Y \Rightarrow_r Z$ with $X \Rightarrow^+ Y$ being the m^{th} dependence chain, $1 \leq m \leq n$, and such that **extend** can be applied. For, by the property $(*)$, the step $G_j \Rightarrow_r G_{j+1}$ removes an item generated by one of the dependence chains and hence cannot be shifted beyond this chain. □

Proof of Theorem 3.12 A graph rewriting system that is not consumptive or that admits an infinite forward closure is clearly non-terminating. Conversely, Lemmas 3.17 and 3.19 show that a non-terminating consumptive system admits an infinite forward closure. □

4 Application to term graph rewriting

In this section the application of Theorem 3.12 to term graph rewriting is sketched. Details of the approch used here can be found in [HP91, Plu93]. (For related approaches, see [BvEG+87, CR93] and the collection [SPvE93].)

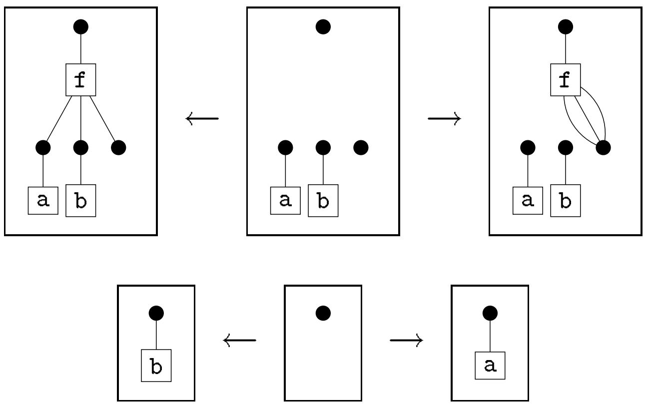
Let Σ be a signature such that $type(f) \geq 1$ for each $f \in \Sigma$. Such an f is considered as function symbol of arity $type(f)-1$ (0-ary symbols are constants). Given a hypergraph G over Σ and a hyperedge e with $type(lab_G(e)) = n$, the nodes at positions 1 to $n-1$ in $att_G(e)$ are the *argument nodes* of e while the node at position n is the *result node*. G is a *jungle* if there are no cycles and if each node is the result node of at most one hyperedge. Jungles represent terms similar to trees but allow shared subterms. For example, the hypergraph in Figure 1 is a jungle representing the term `f(a,b,b)` with a shared constant `b`.

A term rewrite rule $l \to r$ consists of two terms l and r that may contain variables. The variables in r must also occur in l, and l must not be a single variable. Each such rule is translated into a graph rule $(L \leftarrow K \to R)$, where L and R are jungles representing l and r. K is obtained from L by removing the hyperedge representing the leftmost function symbol in l. Variables are represented as non-result nodes, where multiple occurrences of a variable in l or r are always shared in L and R.

Example 4.1 The term rewrite rules

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

(where `x` is a variable) are translated into the following graph rewrite rules:



The graph rules resulting from the translation of term rewrite rules are called *evaluation rules*. The set of all evaluation rules for a given term rewriting system is denoted by $\mathcal{E}$. Evaluation rules preserve the jungle structure; the rewrite relation on jungles induced by $\mathcal{E}$ is denoted by $\Rightarrow_{\mathcal{E}}$.

To test the relation $\Rightarrow_{\mathcal{E}}$ for termination, only forward closures starting from jungles need to be considered since $\Rightarrow_{\mathcal{E}}$ is a relation on jungles. Moreover, evaluation rules are consumptive as each application removes a hyperedge. Hence Theorem 3.12 can be refined as follows, where an *infinite jungle forward closure* is an infinite forward closure starting from a jungle.

Theorem 4.2 *The relation $\Rightarrow_{\mathcal{E}}$ is terminating if, and only if, $\mathcal{E}$ does not admit an infinite jungle forward closure.*

Example 4.3 It is easy to check that the two evaluation rules in Example 4.1 induce only jungle forward closures of at most two steps. There are three one-step forward closures resulting from the first rule (the variable node can be identified with each of the two result nodes of the hyperedges for `a` and `b`) and one one-step forward closure resulting from the second rule. The three former closures can be extended by applying the second rule, yielding the only possible two-step closures. Obviously, these cannot be extended. Hence, by Theorem 4.2, the relation $\Rightarrow_{\mathcal{E}}$ is terminating. In contrast, the given term rewriting system is non-terminating due to the following infinite rewrite sequence:

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

Thus Theorem 4.2 can be used to prove the termination of evaluation rules even if term rewriting is non-terminating (vice versa it holds that $\Rightarrow_{\mathcal{E}}$ is terminating whenever term rewriting is [HP91]).

It is worth noting that the above rules constitute a non-terminating system if they are used on arbitrary hypergraphs. The reason is that the first rule can be applied to the hypergraph that is obtained from the left-hand side by fusing the three argument nodes of the `f`-hyperedge. This application yields the same hypergraph again (which is not a jungle since the two constant hyperedges have the same result node), hence there is a cyclic derivation.

The jungle evaluation relation $\Rightarrow_{\mathcal{E}}$ is not adequate for non-left-linear term rewriting systems, i.e. systems where some rules have repeated variables in their left-hand sides. This is because for such a system, an evaluation rule need not be applicable to a jungle although the underlying term rewrite rule is applicable to the represented term. This problem is overcome in [HP91, Plu93] by adding "folding rules" which fuse hyperedges with identical labels and argument nodes. The termination of systems of evaluation and folding rules can still be characterized as in Theorem 4.2. However, the folding rules cause infinitely many (finite) jungle forward closures. Future research has to show whether this problem can be solved by further restricting the class of jungle forward closures.

References

[BvEG+87] Hendrik Barendregt, Marko van Eekelen, John Glauert, Richard Kennaway, Rinus Plasmeijer, and Ronan Sleep. Term graph rewriting. In *Proc. Parallel Architectures and Languages Europe*, pages 141–158. Springer Lecture Notes in Computer Science 259, 1987.

[CR93] Andrea Corradini and Francesca Rossi. Hyperedge replacement jungle rewriting for term rewriting systems and logic programming. *Theoretical Computer Science*, 109:7–48, 1993.

[Der81] Nachum Dershowitz. Termination of linear rewriting systems (preliminary version). In *Proc. Automata, Languages, and Programming*, pages 448–458. Springer Lecture Notes in Computer Science 115, 1981.

[Der87] Nachum Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3:69–116, 1987.

[Ehr79] Hartmut Ehrig. Introduction to the algebraic theory of graph grammars. In *Proc. Graph-Grammars and Their Application to Computer Science and Biology*, pages 1–69. Springer Lecture Notes in Computer Science 73, 1979.

[EK76] Hartmut Ehrig and Hans-Jörg Kreowski. Parallelism of manipulations in multidimensional information structures. In *Proc. Mathematical Foundations of Computer Science*, pages 284–293. Springer Lecture Notes in Computer Science 45, 1976.

[Geu89] Oliver Geupel. Overlap closures and termination of term rewriting systems. Report MIP-8922, Fakultät für Mathematik und Informatik, Universität Passau, Passau, Germany, 1989.

[Hab92] Annegret Habel. Hypergraph grammars: Transformational and algorithmic aspects. *Journal of Information Processing and Cybernetics*, 5:241–277, 1992.

[HP91] Berthold Hoffmann and Detlef Plump. Implementing term rewriting by jungle evaluation. *RAIRO Theoretical Informatics and Applications*, 25(5):445–472, 1991.

[Kre77] Hans-Jörg Kreowski. Manipulationen von Graphmanipulationen. Dissertation, Technische Universität Berlin, 1977.

[Plu93] Detlef Plump. Evaluation of functional expressions by hypergraph rewriting. Dissertation, Universität Bremen, Fachbereich Mathematik und Informatik, 1993.

[RS85] Neil Robertson and P.D. Seymour. Graph minors — a survey. In Ian Anderson, editor, *Surveys in Combinatorics 1985*, pages 153–171. London Mathematical Society, 1985.

[SPvE93] Ronan Sleep, Rinus Plasmeijer, and Marko van Eekelen, editors. *Term Graph Rewriting: Theory and Practice*. John Wiley, 1993.