# 12

# The Adequacy of Term Graph Rewriting for Simulating Term Rewriting

J.R. Kennaway, J.W. Klop, M.R. Sleep and F.J. de Vries

## 12.1 INTRODUCTION

What does it mean to say that a Graph Rewrite System $(GRAPHS, \rightarrow_{graph})$ is an implementation of a Term Rewrite System $(TERMS, \rightarrow_{term})$? The intuitive answer is (cf. [vEB90]): everything which term rewriting can do can be performed by graph rewriting as well, modulo the unravelling of graphs to terms.

Unfortunately, if we insist on this notion, we soon discover that graph rewriting does *not* implement term rewriting. For example, consider rules such as $D(x) \rightarrow P(x,x), A \rightarrow B, P(A,B) \rightarrow C$. With term rewriting we get the sequence: $D(A) \rightarrow_{term} P(A,A) \rightarrow_{term} P(A,B) \rightarrow_{term} C$, so that $D(A) \rightarrow^* C$ using term rewriting. But with graph rewriting, the rule $D(x) \rightarrow_{graph} P(x,x)$ copies only a pointer to the subterm $x$, and the only possible graph rewriting of $D(A)$ is the sequence $D(A) \rightarrow_{graph} P(A,A) \rightarrow_{graph} P(B,B)$. So, unlike term rewriting, graph rewriting cannot rewrite $D(A)$ to $C$.

With term rewriting, two distinct copies of the argument $x$ of the rule $D(x) \rightarrow P(x,x)$ are made, and can be treated differently in subsequent term rewriting.

With graph rewriting, such duplicating rules copy only pointers, and any subsequent rewrites of the subterm referenced by such a pointer is experienced by every term graph using that pointer. This is good for operational efficiency, but bad if we want to preserve term rewriting-semantics. The main purpose of this chapter is to identify precisely a class of term rewriting systems whose semantics are preserved by graph rewriting implementations.

*Term Graph Rewriting: Theory and Practice.*
Eds. Ronan Sleep, Rinus Plasmeijer and Marko van Eekelen. 

From the above examples, it is clear that term rewriting has more possibilities than graph rewriting. This suggests we treat graph rewriting as a constrained form of term rewriting in which certain term rewriting sequences are prohibited. *Rather than think of graph rewriting as implementing term rewriting, we should instead think of term rewriting implementing graph rewriting.* For certain classes of Term Rewriting System, the implementation works both ways, and there is a precise sense in which we can say that cyclic graph rewriting simulates term rewriting, and which works for a large class of Term Rewriting Systems called *rational orthogonal* systems.

### 12.1.1 Background

Several authors have written on the correctness of graph rewriting implementations of term rewriting [Sta80, BvEG+87, FW91]. Most restrict attention to acyclic graphs and orthogonal rule systems, although Farmer and Watro study the cyclic Y-combinator. In [BvEG+87] it is proved, for any orthogonal Term Rewrite System and its related Graph Rewrite System, that if an acyclic graph g unravels to a term t, then g has a normal form by acyclic graph rewriting if and only if t has a normal form by tree rewriting, and the normal form of g unravels to the normal form of t. There are a number of things one might want to improve in this result.

**Simulate finite approximations to infinite reductions:** the result only applies to terms which have a normal form, but this is in general undecidable. Lazy functional programming is based on finite approximations to infinite sequences, and our theory should deal with these.

**Take advantage of cyclic graphs:** most previous work is restricted to acyclic graph rewriting, but rewriting cyclic graphs has clear practical advantages, in some cases (e.g. for the Y combinator) reducing a transfinite number of rewrites to a finite number.

Cyclic graphs arise naturally as a result of certain optimizations in functional language implementation. Such graphs unravel to infinite terms, and their reduction sequences unravel to transfinite term reduction sequences. Infinitary term rewriting has recently received detailed attention [FW91, DK89, DKP89, DKP91, KKSdV91, KKSdV90a].

### 12.1.2 Overview

In this chapter we give a precise analysis of the relationship between cyclic term graph rewriting and infinitary term rewriting, for orthogonal rewrite systems. For non-orthogonal systems, graph rewriting and term rewriting differ significantly, although some of our results still hold.

Terms and computations of a finite acyclic Graph Rewriting System can be unravelled into terms and computations of a finitary Term Rewriting System. We will give an abstract definition of an adequate mapping between abstract reduction systems which captures the properties of the unravelling mapping. For example, *finite acyclic graph rewriting is adequate for finite term rewriting.*

The main result in [BvEG+87] is a corollary. To extend this to finitary cyclic graph rewriting, we have to consider *infinitary* term rewriting. Abstract reduction systems

form a semantics only for finitary rewriting. By use of the weighted metric abstract reduction systems of [Ken92] as a semantics for infinitary rewriting we strengthen the adequacy concept so that it applies to finitary cyclic graph rewriting.

Our main result is that: *Finite graph rewriting is adequate for rational term rewriting.* In [KKSdV92] we show by means of an counter-example that: *Infinite graph rewriting is not adequate for infinite term rewriting.*

Our definition of an adequate mapping of one system to another adds to the abundance of concepts of simulation, in term rewriting (e.g. [BvEG+87, O'D85]), complexity theory (for an overview see [vEB90]) or programming languages [Mit91].

## 12.2 TERM REWRITING

In [KKSdV91] we develop a theory of infinitary orthogonal term rewriting. General introductions to term rewriting are [DJ90] and [Klo92]. Below we give only key definitions.

### 12.2.1 Infinitary Term Rewriting Systems

An *infinitary Term Rewriting System* over a signature $\Sigma$ is a pair $(Ter^{\infty}(\Sigma), R)$ consisting of the set $Ter^{\infty}(\Sigma)$ of finite and infinite terms over $\Sigma$ and a set of rewrite rules $R \subseteq Ter(\Sigma) \times Ter^{\infty}(\Sigma)$. Note that we require that the left-hand side of a rule is a finite term.

The definition of orthogonality for finitary Term Rewriting Systems extends verbatim to infinitary systems.

### 12.2.2 Strongly converging reductions

The set of terms over a signature can be made into a complete metric space (see for instance [AN80]). The metric (in fact, an ultrametric) is given by $d(t,s) = 0$ if $t$ and $s$ are equal, and is otherwise $1/2^k$, where $k$ is the largest number such that the labels of all nodes of $s$ and $t$ at depth less than or equal to $k$ are equally labeled. Now consider the following rule systems and reduction sequences.

a. $A \to B \to A \to B \to \ldots$, in a TRS with rules $A \to B$ and $B \to A$.
b. $D(E) \to D(S(E)) \to D(S(S(E))) \to \ldots$, in a TRS with rule $D(x) \to D(S(x))$.
c. $C \to S(C) \to S(S(C)) \to \ldots$, in a TRS with rule $C \to S(C)$.

The first example is a diverging reduction sequence. The second is a *weakly converging* reduction with limit $D(S^{\omega})$. The final example is *strongly converging* with limit $S^{\omega}$. The distinction between the two types of convergence is that a weakly converging reduction need only converge in the topological sense, while a strongly converging reduction must satisfy the additional requirement that the depth of reduced redexes tends to infinity.

In [KKSdV91] we have shown that strongly converging transfinite reduction has a more well-behaved theory than weakly converging transfinite reduction. For this reason, we here consider only the former type of transfinite reduction. We write $t \to^{\omega} s$ (resp. $t \to^{\leq\omega} s$) to denote a strongly converging reduction of length $\omega$ (resp. at most

$\omega$) from $t$ to $s$. Concatenation of (a possibly infinite number of) reductions gives reductions of any ordinal length. For such a reduction of length $\alpha$ to be strongly converging we require that, considered as a mapping from $\alpha + 1$ to terms, it be continuous with respect to the usual topology on ordinals, that the depth of reduced redexes tends to infinity, and that every proper initial segment is also strongly converging. (The essential content of the last condition is that the depth of reduced redexes tends to infinity at every limit ordinal $\lambda \leq \alpha$.) We write $t \rightarrow^{\alpha} s$ for a strongly converging reduction of ordinal length $\alpha$. While the notion of a reduction sequence longer than $\omega$ may seem devoid of computational content, the Compressing Lemma of [KKSdV91] shows that if $t$ reduces to $s$ by a strongly converging reduction of length greater than $\omega$, it also reduces to $s$ by a sequence of length at most $\omega$. Finally, $t \rightarrow^{\infty} s$ denotes a strongly converging reduction of any finite or infinite length. Note that it is easily shown that the length is at most countable, and can be any countable ordinal.

## 12.3 GRAPH REWRITING

Graph rewriting is a common method of implementing term rewrite languages [Pey87]. It relies on the basic insight, that when a variable occurs many times on the right-hand side of a rule, one need only copy pointers to the corresponding parts of the term being evaluated, instead of making copies of the whole subterm. The reader familiar with graph rewriting may skip this section. Note however that we allow cyclic graphs; these correspond to certain infinite terms.

DEFINITION **12.3.1** *A term graph $g$ over a signature $\Sigma = (F, V)$ is a quadruple $(nodes(g), lab(g), succ(g), roots(g))$, where $nodes(g)$ is a finite or infinite set of nodes, $lab(g)$ is a function from a subset of the nodes of $g$ to $F$, $succ(g)$ is a function from the same subset to tuples of nodes of $g$, and $roots(g)$ is a tuple of (not necessarily distinct) nodes of $g$. Furthermore, every node of $g$ must be accessible (defined below) from at least one root. Nodes of $g$ outside the common domain of $lab(g)$ and $succ(g)$ are called empty.*

DEFINITION **12.3.2** *A path in a graph $g$ is a finite or infinite sequence $a, i, b, j, \ldots$ of alternating nodes and integers, beginning and (if finite) ending with a node of $g$, such that for each $m, i, n$ in the sequence, where $m$ and $n$ are nodes, $n$ is the $i^{th}$ successor of $m$. The length of the path is the number of integers in it. If the path starts from a node $m$ and ends at a node $n$, it is said to be a path from $m$ to $n$. If there is a path from $m$ to $n$, then $n$ is said to be accessible from $m$. When this is so, the distance of $n$ from $m$ is the length of a shortest path from $m$ to $n$.*

We may write $n : F(n_1, \ldots, n_k)$ to indicate that $lab(g)(n) = F$ and $succ(g)(n) = (n_1, \ldots, n_k)$. A finite graph may then be presented as a list of such *node definitions.*

For example, the list of node definitions $x : F(y, z), z : G(y, w, w), w : H(w)$ represents the graph shown in figure 12.1.

In such pictures, we may omit the names $x, y, z, \ldots$ as their only function in the textual representation is to identify the nodes. In particular, $x, y, z, \ldots$ do not represent variables: variables are represented by empty nodes. Different empty nodes need only be distinguished by the fact that they are different nodes; we do not

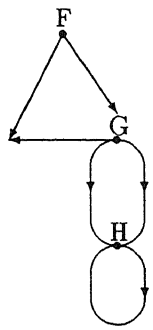


**Figure 12.1** The graph $x : F(y, z), z : G(y, w, w), w : H(w)$

need any separate alphabet of variable names. Multiple references to the same variable in a term are represented in a graph by multiple references to the same empty node. The tabular description demonstrated above may conveniently be condensed, by nesting the definitions; for example, another way of writing the same graph is $F(y, z : G(y, w, w : H(w)))$. In general a graph may have more than one root. We will only use graphs with either one root (such graphs represent terms) and graphs with two roots (which represent term rewrite rules).

DEFINITION **12.3.3** *A graph homomorphism from a graph $g$ to a graph $g$ is a function $f$ from the nodes of $g$ to the nodes of $g$, such that for all nodes $n$ in the domain of $lab(g)$, $lab(h)(f(n)) = lab(g)(n)$, and $succ(h)(f(n)) = succ(g)(n)$.*

Note that a graph homomorphism is not required to map the roots of its domain to the roots of its codomain. On graphs one can define many general graph rewrite mechanisms. We are concerned with one particular form: term graph rewriting.

DEFINITION **12.3.4** *A term graph is a graph with one root.*

*A term graph rewrite rule is a graph with two, not necessarily distinct, roots (called the left and right roots), in which every empty node is accessible from the left root, and the subgraph containing those nodes accessible from the left root is a finite tree. The left- (resp. right-) hand side of a term graph rewrite rule $r$ is the subgraph consisting of all nodes and edges accessible from the left (resp. right) root: notation $left(r)$ (resp. $right(r)$).*

*A redex of a term graph rewrite rule $r$ in a graph $g$ is a homomorphism from the left-hand side of $r$ to $g$. The occurrence of the redex is the minimal occurrence of the node of $g$ to which the left root is mapped. The depth of a redex is the length of the occurrence.*

DEFINITION **12.3.5** *The result of reducing a redex of the rule $r$ in a graph $g$ at occurrence $u$ is the graph obtained by the following construction.*

*a. Construct a graph $h$ by adding to $g$ a copy of all nodes and edges of $r$ not in $left(r)$. Where such an edge has one endpoint in $left(r)$, the copy of that edge in $h$ is connected to the image of that endpoint by the homomorphism.*

*b. Let $n_l$ be the node of $h$ corresponding to the left root of $r$, and $n_r$ the node corresponding to the right root of $r$. (These are not necessarily distinct.) In $h$, replace*

*every edge whose target is $n_l$ by an edge with the same source and target $n_r$, obtaining a graph $k$. The root of $k$ is the root of $h$, unless this is $n_l$, otherwise it is $n_r$.*

c. *Remove all nodes which are not accessible from the root of $k$. The resulting graph is the result of the rewrite.*

We have now the ingredients to give the general definition of a Term Graph Rewrite System.

DEFINITION **12.3.6** *Let $\Sigma$ be a signature. A Term Graph Rewrite System (GRS for short) is a pair $(G(\Sigma), R)$ where $G(\Sigma)$ is the set of graphs for the signature $\Sigma$, and $R$ a set of term graph rewrite rules for the signature $\Sigma$.*

Having defined term graph rewriting and the notion of depth on term graphs, the concepts of normal form, infinitary rewriting, orthogonality, etc. carry over to term graphs.

As an example consider the rule $I(x) \rightarrow x.$ and the graph $n : I(n)$ shown in figure 12.2.

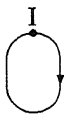


**Figure 12.2** The circular term graph $x : I(x)$

The graph $n : I(n)$ corresponds to the infinite term$I^\omega = I(I(I(\ldots)))$, and is a redex of the rule. This "circular I" graph reduces to itself according to Definition 12.3.5 above. Circular I is one instance of a class of redexes having the same behaviour, the *circular redexes.*

DEFINITION **12.3.7** a. *A redex of a rule $r$ is circular if the roots of $r$ are distinct and the homomorphism from $left(r)$ to $g$ maps both roots of $r$ to the same node. (This can only happen if the right root of $r$ is accessible from the left root.)*

b. *A rule is a collapse rule if its right root is a variable.*

An example of a collapsing rule is $x : Head(Cons(y, z)) \rightarrow y$. An example of a non-collapsing rule which admits circular redexes is $x : F(y : F(z)) \rightarrow y$. Note that this rule conflicts with itself: it has two overlapping redexes in the graph F(F(F(G))). A circular redex of this rule is x:F(x).

PROPOSITION **12.3.8** *[KKSdV90a] In an orthogonal Term Graph Rewriting System, a rule has a circular redex if and only if it is a collapse rule.*

From now on we will consider term graphs and term graph rewriting only, and often we will simply call them graphs and graph rewriting.

### 12.3.1 Unravelling

Unravelling transforms (term) graphs to terms. Both graphs and computations can be unravelled. In [KKSdV92] we show that, for any Graph Rewrite System, if $g$ reduces to $g'$ by strongly convergent reduction, then $U(g)$ similarly reduces to $U(g')$ in the unravelled system. This is so even for non-orthogonal systems. Below we state key definitions and results.

DEFINITION **12.3.9** *The unravelling $U(g)$ of a graph $g$ is the term representation of the following forest. The nodes of $U(g)$ are the paths of $g$ which start from any of its roots. Given a node $a, i, b, j, \ldots, y$ of $U(g)$, if $y$ is a nonempty node of $g$, then this node of $U(g)$ is labeled with the function symbol $lab(g)(y)$, and its successors are all paths of the form $a, i, b, j, \ldots, y, n, z$, where $z$ is the nth successor of $y$ in $g$. If $y$ is empty, then it is labeled with a variable symbol, a different symbol being chosen for every empty node of $g$.*

Note that a cyclic graph will have an infinite unravelling. For example, the unravelling of the graph shown in the previous picture is the term $F(y, G(y, H^\omega, H^\omega))$.

It is easy to see that for a term graph $g$, $U(g)$ is a term, and for a graph rewrite rule $r$, $U(r)$ is a term rewrite rule. We can also apply the notion of unravelling to a whole rewrite system.

DEFINITION **12.3.10** *The unravelling of a Graph Rewriting System $(G(\Sigma), R))$ is the Term Rewriting System $(Ter^\infty(\Sigma), U(R))$ whose rules $U(R)$ are the unravellings of the rules in $R$. This TRS is also denoted by $U(G(\Sigma), R)$; its set of terms is $U(G)$.*

So, given a signature $\Sigma$ the operator $U$ transforms GRSs over $\Sigma$ into TRSs over $\Sigma$. Note that a GRS is orthogonal if and only if its unravelling is orthogonal.

PROPOSITION **12.3.11** *There is a homomorphism from $U(g)$ to $g$ which takes the root of $U(g)$ to the root of $g$.*

The homomorphism is obtained by mapping each node of $U(g)$ (which is a finite path of $g$) to its final element. If $g$ is acyclic, this is clearly the only homomorphism from $U(g)$ to $g$, but if g is cyclic there can be more than one: for example, if $g = x : A(A(x))$, there are two.

PROPOSITION **12.3.12** *A graph $g$ in the GRS $(G(\Sigma), R)$ is a normal form if and only if its unravelling $U(g)$ is a normal form in $(Ter^\infty(\Sigma), U(R))$.*

THEOREM **12.3.13** *Let $g \to g'$ in a GRS. Then $U(g) \to_{\leq\omega} U(g')$ in the corresponding TRS. Moreover, the depth of every redex reduced in the term sequence is at least equal to the depth of the redex reduced in $g$.*

COROLLARY. Let $g \to_\alpha g'$ in a GRS for some infinite ordinal $\alpha$ Then $U(g) \to_{\leq\alpha} U(g')$ in the corresponding TRS.

### 12.3.2 Lifting.

Lifting transforms a Term Rewrite System into a Graph Rewrite System. We chose here a very simple lifting from the many GRS which unravel to the given TRS. We transform a term rule into a graph rewrite rule by sharing multiple occurrences of a variable in the right-hand side. This version is compatible with the one in [BvEG+87]. In contrast to unravelling, it does not have nice reduction preserving properties.

DEFINITION **12.3.14** *The lifting $L(Ter^{\omega}(\Sigma), R)$ of the TRS $L(Ter^{\omega}(\Sigma), R)$ is defined as the GRS $(G(\Sigma), R)$ where the elements of $R$ are minimally shared, bi-rooted graphs, corresponding with the rules in $R$: reading the left- and right-hand sides of a term rule $T \to T'$ as trees, and then for each variable identifing all leaves of the two trees which bear that variable. The roots of the two trees become the roots of the graph.*

An example is provided by the following TRS:

$F(x) \to A(x, x)$
$A(D, D) \to B$
$C \to D$

The term $F(C)$ can reduce in the following way:

**Term rewriting:** $F(C) \to A(C, C) \to A(C, D) \to A(D, D) \to B$
**Graph rewriting:** $F(C) \to A(C, C) \to A(D, D) \to B$

Note that this example shows that there is not an exact counterpart of theorem 12.3.13 for lifting. Although $F(C)$ reduces to $A(C, D)$, the graph $F(C)$ in the lifted Term Rewrite System does not reduce to the graph $A(C, D)$, but reduces from $A(C, C)$ to $A(D, D)$ in a single step..

## 12.4 ADEQUACY: A PRECISE NOTION OF SIMULATION

Recall from the introductory remarks that graph rewriting may be viewed as a restricted form of term rewriting, and that while term rewriting can simulate graph rewriting, the reverse is not in general true. We want to develop some notion of simulation which captures some sense in which graph rewriting does simulate term rewriting.

Somewhat counter-intuitively, we start by writing down sensible conditions for a TRS to simulate a GRS. We will then strengthen these conditions to introduce some senses in which the GRS simulates a TRS.

If the Term Rewriting System is to simulate the Graph Rewriting System, the following requirements seems reasonable:

a. For every $g \in G$ there must be some term $U(g) \in T$.
b. If the graph $g \in G$ is a normal form, then the term $U(g)$ is also a normal form. Thus whenever the GRS reaches a dead end, so does the TRS.
c. Whenever $g \to_G g'$, then $U(g) \to^*_T U(g')$. That is, for every (single step) graph rewrite, there is a (perhaps multi-step) term rewriting sequence.

These three conditions allow a graph rewrite sequence to be simulated by a term rewriting sequence containing more steps, but this is in accord with the idea that the TRS is a "machine with finer grain instructions" than the GRS: that is, it can take the TRS several steps to simulate a single GRS step.

We now strengthen the three conditions so as to introduce some sort of notion of simulation of the TRS by the GRS:

a. For every $g \in G$ there must be some term $t \in T$. *Additionally require that $U$ is surjective, so that there is some $g \in G$ for every $t \in T$.*
b. If the graph $g \in G$ is a normal form, then the term $U(g)$ is also a normal form. *Require the converse, as well, so that whenever $U(g)$ is a normal form, so is $g$.*
c. We have seen that because a TRS is finer grain than a corresponding GRS, there are some TRS sequences which have no corresponding GRS sequence. *However, we can often extend such TRS sequences, by further term rewriting, to a point where there is a corresponding graph sequence.* This suggests the following modified condition: *Whenever $U(g) \rightarrow^*_{term} t'$, then there exist $t'' \in T$ and $g'' \in G$ such that $t' \rightarrow^*_{term} t''$ and $g \rightarrow^*_{graph} g''$.*

The above discussion motivates the following definition of an adequate mapping:

DEFINITION **12.4.1** *A mapping $U(G, \rightarrow) \rightarrow (T, \rightarrow)$ is an adequate mapping if:*

a. *$U$ is surjective.*
b. *$g \in G$ is a normal form if and only if $U(g)$ is a normal form.*
c. *For $g \in G$, if $U(g) \rightarrow^*_{term} t'$ then there is a $g'' \in G$ such that $g \rightarrow^*_{graph} g''$ and $t' \rightarrow^*_{term} U(g'')$.*

One test of the reasonableness of this definition is to check that term rewritings such as $t \rightarrow^*_{term} t_{nf}$ to normal form can be simulated by (adequate) graph rewriting:

a. find an element $g$ of $G$ such that $U(g) = t$. This is guaranteed by condition (a).
b. now term rewrite $t$ to normal form $t_{tnf}$. By condition (c) there is a $g''$ such that $g$ graph rewrites to $g''$ and $t_{tnf}$ term rewrites to $U(g'')$. As $t_{tnf}$ is a normal form, this means $t_{tnf} = U(g'')$, and by (b) $g$" is a normal form.

So, if there is a normalizing sequence for $t$ by term rewriting, there is a corresponding normalizing sequence by graph rewriting. But the adequacy condition can do better than this. Let $t \rightarrow^* t'$ be a finite or infinite term rewriting sequence. Then, by condition (c), we can extend the term rewriting sequence to obtain $t \rightarrow^* t' \rightarrow^* t''$, such that there are $g, g''$ such that $g$ graph rewrites to $g''$, and $t = U(g), t'' = U(g'')$. This means that not only can we simulate term rewritings to normal form, but also that we can simulate transfinite term rewriting sequences which develop increasing approximations to an infinite term. Such notions are common in lazy functional languages.

Our "adequacy" notion is one of many such notions of simulation, implementation etc., of one rewrite system in another, which we have encountered in the literature. The notions are compared in [KKSdV92].

## 12.5 ADEQUACY OF ACYCLIC GRAPH REWRITING FOR FINITE TERM REWRITING

We first state a result which is well known. Our proof is used as a basis for our more general result on cyclic graph rewriting and rational term rewriting. It depends on the notion of a complete development in an orthogonal rewriting system. Given a set of redexes of a term, a complete development of that set is a reduction sequence in which each step is the reduction of some residual of a member of the set, ending with a term containing no such residuals. A *Gross-Knuth* reduction of a term t is a complete development of all the redexes in t. We denote such a reduction sequence by $GKS(t)$ and its final term by $GK(t)$. Gross-Knuth reduction is similarly defined for graphs.

For finite terms and acyclic graphs, every set of redexes has a complete development. Note that while there may be many different orders in which the redexes may be reduced, the final result of a complete development of a given set of redexes is independent of the order. For infinite terms and cyclic graphs it is more complicated, as we shall see in the next section.

THEOREM **12.5.1** *Finite orthogonal acyclic graph rewriting is adequate for finite orthogonal term rewriting, via the unravelling mapping.*

PROOF. The first two adequacy conditions are trivially satisfied. For the cofinality condition, let there be given a reduction sequence $t_0 \to t_1 \to \ldots$, where $t_0 = U(g_0)$. For $i \geq 0$ let $r_i$ be the one-step sequence from $t_i$ to $t_{i+1}$. Construct a term reduction diagram and a graph reduction sequence inductively as follows.

$t_0'' = t_0$.
$T_0'' = r_0$.
$T_i'' = r_i/(T_i \cdot T_i') : t_i'' \to^* t_{i+1}'$ (when $i \geq 1$).
$t_1' = t_1$.
$T_1 = \langle\rangle : t_1 \to^* t_1'$.
$T_i = (T_{i-1} \cdot T_{i-1}')/r_{i-1} : t_i \to^* t_i'$ (when $i \geq 2$).
$T_i' = T_{i-1}''/(GKS(t_{i-1}'')/T_{i-1}'') : t_i' \to^* t_i''$ (when $i \geq 1$).
$g_i = GK(g_{i-1})$ (when $i \geq 1$).

It is then easy to establish that $t_i'' = U(g_i)$ for all $i \geq 1$, and hence that each $t_i$ reduces to $U(g_i)$, by the sequence $T_i \cdot T_i'$. This proves cofinality. □

Here is an example of how the theorem fails for non-orthogonal rule systems.

Rules: $F(x) \to A(x,x), B \to C, B \to D$.

Term reduction sequence: $F(B) \to A(B,B) \to A(C,B) \to A(C,D)$.

By graph reduction, $F(B)$ can be reduced only to $A(x : B, x)$, $A(x : C, x)$, or $A(x : D, x)$. However, the term $A(C,D)$ cannot be reduced to the unravellings of any of these graphs.

Even for orthogonal systems, the adequacy relation fails for infinite terms and graphs. There exists a rewrite system containing an infinite graph $g$ and an infinite reduction $U(g) \to^\infty t$, such that there is no graph $g'$ for which $g \to^\infty g'$ and $t \to^\infty U(g')$. An example is given in [KKSdV92].

## 12.6 ADEQUACY OF FINITE GRAPH REWRITING FOR RATIONAL TERM REWRITING

Despite the failure of adequacy for infinite rewriting, we want to demonstrate the adequacy of cyclic graph rewriting for the transfinite term rewriting which cyclic graphs intuitively give rise to. We have mentioned that unrestricted transfinite rewriting does not allow such a result. Instead, we consider a restricted version of transfinite term rewriting which corresponds to finite cyclic graph rewriting: *rational* term rewriting.

DEFINITION **12.6.1** *A rational term is a term containing only finitely many non-isomorphic subterms. It can be shown (see [KKSdV92]) that a term is rational if and only if it is the unravelling of a finite graph. A rational set of nodes of a rational term is a set of nodes such that, if each of the nodes in the set is marked, the resulting term is still rational, taking the marks into account when testing isomorphism. A rational set of redexes of a rational term is a set of redexes whose roots are a rational set of nodes.*

THEOREM **12.6.2** *A set of nodes of a rational term $t$ is rational if and only if there is a graph $g$ unravelling to $t$, and a set of nodes of $g$ which map by the unravelling to the given set of nodes of $t$.*

See [KKSdV92] for proof.

DEFINITION **12.6.3** *The rational term reduction sequences are defined by the following axioms:*

*a. A strongly convergent complete development, of length at most $\omega$, of a rational set of redexes, is rational.*
*b. A concatenation of finitely many rational reduction sequences is rational.*
*c. A subsequence of a rational reduction sequence is rational.*
*d. There are no other rational reduction sequences.*

DEFINITION **12.6.4** *A* weakly collapsing *set of rewrite rules is a set which includes at most one rule whose right hand side is a variable, and such that that rule, if present, has the form $A(x) \to x$ for some function symbol $A$.*

THEOREM **12.6.5** *In weakly collapsing rewrite systems, finitary (cyclic) graph rewriting is adequate for rational infinitary term rewriting.*

We shall only give an idea of the proof here. It is easy to demonstrate that every rational reduction sequence is a finite concatenation of complete developments of rational sets of redexes. We imitate the proof of the adequacy theorem for acyclic graphs and finitary term rewriting, but in which each step $t_i \to t_{i+1}$ is a complete development of a rational set of redexes. The corresponding reduction from $g_i$ to $g_{i+1}$ is then the complete development of all redexes of $g_i$ which are mapped by unravelling to the redexes reduced in $t_i$. The weakly collapsing condition on the system ensures that the result of such a complete development is independent of its order, as shown in [KKSdV92]. The necessity of this condition is illustrated by the following example.

Rules: $A(x) \to x$, $B(x) \to x$

Graph: $x : A(B(x))$

The Church-Rosser property fails here. The graph can reduce to either $x : A(x)$ or to $x : B(x)$, neither of which can reduce to anything but itself. Failure of the Church-Rosser property immediately leads to a failure of the adequacy property. Consider the graph $F(x,x), x : A(B(x))$. This reduces only to $F(x,x), x : A(x)$ or $F(x,x), x : B(x)$. Its unravelling is the infinite term $F(A(B(A(B(\ldots)))), A(B(A(B(\ldots)))))$. This can be reduced by a rational reduction sequence to $F(A(A(A(\ldots))), B(B(B(\ldots))))$. This term cannot be further reduced to the unravelling of either $F(x,x), x : A(x)$ or $F(x,x), x : B(x)$, contradicting the adequacy property.

The weakly collapsing condition is not as restrictive as it might at first appear. Every non-weakly collapsing system can be transformed into a weakly collapsing one by adding a new rule $I(x) \to x$, and replacing every right-hand side which is a variable $x$ by $I(x)$. This transformation is closely similar to the way such rules are implemented in practice.

## REFERENCES


[AGM92] S. Abramsky, D. Gabbay, and T. Maibaum (editors). *Handbook of Logic in Computer Science*, vol. II. Oxford University Press, 1992.

[AN80] A. Arnold and M. Nivat. The metric space of infinite trees: Algebraic and topological properties. *Fundamenta Informatica*, **4**, pp. 445–476, 1980.

[Boo91] R. V. Book (editor). *Proc. 4th Conference on Rewriting Techniques and Applications*, Springer Verlag, Lecture Notes in Computer Science 488, Como, Italy, 1991.

[BvEG+87] H. P. Barendregt, M. C. J. D. van Eekelen, J. R. W. Glauert, J. R. Kennaway, M. J. Plasmeijer, and M. R. Sleep. Term graph rewriting. In J. W. de Bakker, A. J. Nijman, and P. C. Treleaven (editors), *Proc. PARLE'87 Conference, vol.II*, Springer Verlag, Lecture Notes in Computer Science 259, pp. 141–158, Eindhoven, The Netherlands, 1987.

[DJ90] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In van Leeuwen [vL90a], chapter 15.

[DK89] N. Dershowitz and S. Kaplan. Rewrite, rewrite, rewrite, rewrite, rewrite. In *Proc. ACM Conference on Principles of Programming Languages, Austin, Texas*, pp. 250–259, Austin, Texas, 1989.

[DKP89] N. Dershowitz, S. Kaplan, and D. A. Plaisted. Infinite normal forms (plus corrigendum). In G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi Della Rocca (editors), *Automata, Languages and Programming*, Springer Verlag, Lecture Notes in Computer Science 372, pp. 249–262, Stresa, Italy, 1989.

[DKP91] N. Dershowitz, S. Kaplan, and D. A. Plaisted. Rewrite, rewrite, rewrite, rewrite, rewrite. *Theoretical Computer Science*, **83**, pp. 71–96, 1991. Extended version of [DKP89].

[FW91] W. M. Farmer and R. J. Watro. Redex capturing in term graph rewriting. In Book [Boo91], pp. 13–24.

[Ken92] J. R. Kennaway. *On transfinite abstract reduction systems.* Technical Report CS-R9205, Centre for Mathematics and Computer Science, Amsterdam, 1992.

[KKSdV90a] J. R. Kennaway, J. W. Klop, M. R. Sleep, and F. J. de Vries. *An Infinitary Church-Rosser property for Non-collapsing Orthogonal Term Rewriting Systems.* Technical Report CS-9043, CWI, Amsterdam, 1990.

[KKSdV90b] J. R. Kennaway, J. W. Klop, M. R. Sleep, and F. J. de Vries. *Transfinite Reductions in Orthogonal Term Rewriting Systems.* Technical Report CS-R9041, CWI, Amsterdam, 1990.

[KKSdV91] J. R. Kennaway, J. W. Klop, M. R. Sleep, and F. J. de Vries. Transfinite reductions in orthogonal term rewriting systems (extended abstract). In Book [Boo91], pp. 1–12. A longer version appears in [KKSdV90b].

[KKSdV92] J. R. Kennaway, J. W. Klop, M. R. Sleep, and F. J. de Vries. *On the adequacy of graph rewriting for simulating term rewriting.* Technical Report CS-R9204, Centre for Mathematics and Computer Science, Amsterdam, 1992.

[Klo92] J. W. Klop. Term rewriting systems. In Abramsky et al. [AGM92], pp. 1–116.

[Mit91] J. C. Mitchell. On abstraction and the expressive power of programming languages. In *Proc. Int. Conf. on Theoretical aspects of Computer Software*, Sendai, Japan, 1991.

[O'D85] M. J. O'Donnell. *Equational Logic as a Programming Language.* MIT Press, 1985.

[Pey87] S. L. Peyton Jones. *The implementation of functional programming languages.* Prentice-Hall, 1987.

[Sta80] J. Staples. Computation on graph-like expressions. *Theoretical Computer Science*, **10**, pp. 171–185, 1980.

[vEB90] P. van Emde Boas. Machine models and simulations. In van Leeuwen [vL90b], chapter 1.

[vL90a] J. van Leeuwen (editor). *Handbook of Theoretical Computer Science*, vol. B: Formal Models and Semantics. North-Holland, Amsterdam, 1990.

[vL90b] J. van Leeuwen (editor). *Handbook of Theoretical Computer Science*, vol. A: Algorithms and Complexity. North-Holland, Amsterdam, 1990.