INFORMATIQUE THÉORIQUE ET APPLICATIONS

BERTHOLD HOFFMANN
DETLEF PLUMP

Implementing term rewriting by jungle evaluation

Informatique théorique et applications, tome 25, n° 5 (1991), p. 445-472.

<http://www.numdam.org/item?id=ITA_1991__25_5_445_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 25, n° 5, 1991, p. 445 à 472)

IMPLEMENTING TERM REWRITING BY JUNGLE EVALUATION (*)

by Berthold HOFFMANN (1) and Detlef PLUMP (1)

Communicated by W. BRAUER

Abstract. – *Jungles are acyclic hypergraphs which represent sets of terms such that common subterms can be shared. Term rewrite rules are translated into jungle evaluation rules which implement parallel term rewriting steps. By using additional hypergraph rules which "fold" equal subterms, even non-left-linear term rewriting systems can be implemented. As a side effect, these folding rules can speed up the evaluation process considerably. It is shown that terminating term rewriting systems result in terminating jungle evaluation systems which are capable to normalize every term. Moreover, confluent and terminating term rewriting systems give rise to confluent and terminating jungle evaluation systems, provided that the "garbage" produced by the evaluation steps is ignored.*

Résumé. – *Les jungles sont des hypergraphes qui représentent des ensembles de termes partageant des sous-termes communs. Les règles de réécriture sont traduites en règles d'évaluation de jungles qui implémentent des étapes de réécriture parallèle de termes. En utilisant des règles supplémentaires d'hypergraphes qui « replient » les sous-termes égaux, il est aussi possible d'implémenter des systèmes de réécriture qui ne sont pas linéaires à gauche. Ces règles de pliage ont comme effet secondaire d'accélérer considérablement le processus. On montre que les systèmes de réécriture à terminaison finie définissent des systèmes d'évaluation de jungles à terminaison finie capables de normaliser chaque terme. De plus, les systèmes de réécriture confluents et à terminaison finie donnent lieu à des systèmes d'évaluation de jungles qui sont de même, confluents et à terminaison finie, à condition d'ignorer les « détritus » produits par les étapes de l'évaluation.*

1. INTRODUCTION

Term rewriting is an interesting way of "computing by replacement" which is used in various areas of computing science: for the interpretation of functional and logical programming languages, for theorem proving, and for

(*) Received January 1989, revised November 1990.
This work is partly supported by the Commission of the European Communities under Contract 390 (PROSPECTRA Project) in the ESPRIT Programme.
(1) Fachbereich Mathematik und Informatik, Universität Bremen, Postfach 330 440, D-2800 Bremen 33.

Informatique théorique et Applications/Theoretical Informatics and Applications
0988-3754/91/05 445 28/$4.80/

executing algebraic specifications of abstract data types and checking properties of specifications like consistency and completeness (*see, e.g.*, [Klo90] and [DJ90] for surveys on term rewriting).

Classically, terms are represented by *trees* and rewriting is realized by *subtree replacement*. Unfortunately, this way of rewriting may be very expensive, both in time and space: the application of a rule may require large subterms to be copied, and each copy of a term has to be rewritten anew.

In this paper we investigate an improved model for implementing term rewriting where this source of inefficiency is avoided:

- Rather than by trees, terms are represented by acyclic hypergraphs so that multiple occurrences of terms can be shared. These hypergraphs are called *jungles*, a name coined in [Plu86] and [HKP88].
- Rewriting is performed by hypergraph replacement, specified by *evaluation rules* according to the algebraic theory of graph grammars (*see*, e. g., [Ehr79]). Instead of copying subterms, evaluation rules create a sharing of subterms.
- Additional hypergraph rules which "fold" equal subterms can be used to achieve a complete sharing of equal subterms; so multiple evaluation of the same term can be avoided.

1.1. Example (Fibonacci numbers)

Consider the term rewrite rules

$$\begin{aligned} \mathrm{fib}(0) &\to 0 \\ \mathrm{fib}(\mathrm{succ}(0)) &\to \mathrm{succ}(0) \\ \mathrm{fib}(\mathrm{succ}(\mathrm{succ}(n))) &\to \mathrm{fib}(\mathrm{succ}(n)) + \mathrm{fib}(n) \end{aligned}$$

specifying a function fib that computes Fibonacci Numbers, based on natural numbers with the constant 0, successor function succ, and addition $+$.

The first two steps for computing the Fibonacci Number of 4 by term rewriting are

$$\begin{aligned} & \mathrm{fib}(\mathrm{succ}^4(0)) \\ \to\ & \mathrm{fib}(\mathrm{succ}^3(0)) + \mathrm{fib}(\mathrm{succ}^2(0)) \\ \to\ & \mathrm{fib}(\mathrm{succ}^2(0)) + \mathrm{fib}(\mathrm{succ}(0)) + \mathrm{fib}(\mathrm{succ}^2(0)) \end{aligned}$$

In both steps, the arguments of fib are copied. In particular, the resulting term contains two copies of $\mathrm{fib}(\mathrm{succ}^2(0))$; each of them must be rewritten anew. As a consequence, rewriting a term $\mathrm{fib}(\mathrm{succ}^n(0))$ to normal form

requires time and space exponential in n (we just consider the three rules shown above; rules defining the addition are neglected).

Figure 1 shows two corresponding jungle evaluation steps, where the arguments are shared instead of being copied. A twofold folding step identifies the nodes representing fib(succ^2(0)), so this subterm has to be evaluated only once (dashed boxes indicate "garbage").

By performing evaluation and folding steps in this order, the evaluation of a term fib(succ^n(0)) requires only a number of steps and space linear in n. □

The first graph grammar approach to the evaluation of functional expressions has been undertaken by Ehrig, Rosen, and Padawitz ([ER76], [Pad82]). However, they use an extension of the algebraic theory of graph grammars having the drawback that most of the results for "standard graph grammars" are not applicable. The graph reduction approaches of Staples [Sta80], Raoult [Rao84] and Barendregt *et al.* [BvEG*87] use formalisms of graph rewriting for which no such comprehensive theory as for the algebraic approach is available.

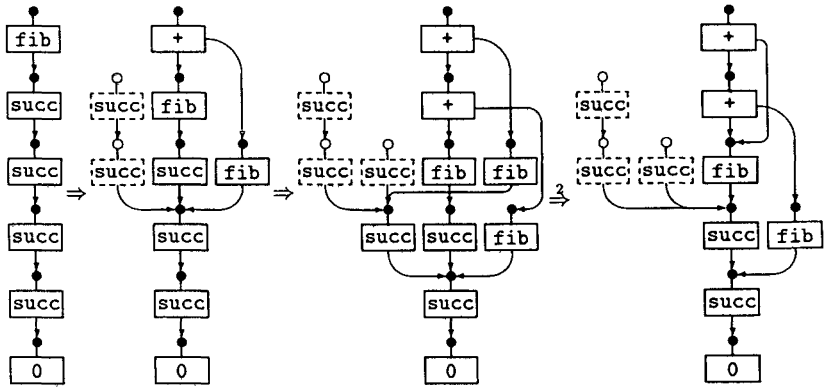


Figure 1. – Jungle evaluation steps followed by a twofold folding step.

In [Hof83] and [Plu86], the authors started an approach to model term rewriting by standard graph grammars. H.-J. Kreowski, A. Habel, and D. Plump continued this work in [HKP88], where hypergraphs are used instead of graphs, in order to make the technical treatment easier.

With this paper we carry forth this research by using a slightly extended class of hypergraph grammars which allows for a translation of term rewrite

rules into hypergraph rules without need for "indirect pointers". We show how jungle evaluation can be used to compute term normal forms, and investigate conditions for termination and confluence of jungle evaluation.

The paper is organized as follows:

We recall basic notions of term rewriting in section 2. In section 3, we define jungles and study their relationship to terms. We discuss properties of morphisms between jungles, and we consider special jungles in order to give sufficient criteria for the existence of such morphisms. Rules for folding multiple representations of terms are introduced in section 4. They allow to compute minimal jungle representations for sets of terms. In section 5 we construct jungle evaluation rules, and show that these rules perform term rewriting. The computation of term normal forms by jungle evaluation is the subject of section 6. In section 7, it is demonstrated that termination of term rewriting carries over to jungle evaluation. Section 8 is devoted to the investigation of confluence criteria for jungle evaluation. Finally, in section 9 we summarize our results, compare our approach with that of Barendregt *et al.*, and point out possible directions of future research.

2. PRELIMINARIES

STRINGS. — A^* denotes the set of all strings over some set A, including the empty string λ. $f^*: A^* \to B^*$ denotes the homomorphic extension of a function $f: A \to B$.

ABSTRACT REDUCTIONS. — Let $\to$ be a binary relation on some set A.

We write $\overset{+}{\to}$ and $\overset{*}{\to}$ for the transitive and transitive-reflexive closure of $\to$, respectively. The n-fold composition of $\to$ (for $n \geqq 0$) is denoted by $\overset{n}{\to}$; in particular, $\overset{0}{\to}$ is the equality on A.

Some $a \in A$ is a *normal form* (with respect to $\to$) if there is no $b \in A$ with $a \to b$. A normal form a is called a *normal form of* $b \in A$ if $b \overset{*}{\to} a$.

We say $\to$ is *terminating* if there is no infinite chain $a_0 \to a_1 \to a_2 \to \ldots$

The relation $\to$ is *confluent* if for all a, b_1, $b_2 \in A$, $b_1 \overset{*}{\leftarrow} a \overset{*}{\to} b_2$ implies $b_1 \overset{*}{\to} c \overset{*}{\leftarrow} b_2$ for some $c \in A$.

SIGNATURES AND TERMS. – Let S be a set and $(\mathrm{OP}_{w,s})_{w\in S^*, s\in S}$ be a family of pairwise disjoint sets. Then $\mathrm{SIG}=(S, \mathrm{OP})$ is a *signature* where $\mathrm{OP}=\bigcup_{w\in S^*, s\in S} \mathrm{OP}_{w,s}$. The elements of S and OP are called *sorts* and *operation symbols*, respectively. An operation symbol $\mathrm{op}\in\mathrm{OP}_{w,s}$ is also written $\mathrm{op}: w\to s$.

Let $(X_s)_{s\in S}$ be a family of pairwise disjoint sets. Then $X=\bigcup_{s\in S} X_s$ is a *variable set* provided that $X\cap \mathrm{OP}=\varnothing$.

Let $s\in S$. A string $t\in(\mathrm{OP}\cup X)^*$ is a *term of sort* s *(over* X*)* if

(i) $t\in X_s$ or

(ii) $t=\mathrm{op}\, t_1 \ldots t_k$ where $\mathrm{op}\in\mathrm{OP}_{s_1\ldots s_k, s}$ and t_i is a term of sort s_i for $i=1, \ldots, k$, $k\geqq 0$.

$T_{\mathrm{SIG}}(X)$ denotes the set of all terms over X. We write $\mathrm{sort}(t)$ for the sort of a term t.

SUBSTITUTIONS. – Let X, Y be variable sets. A function $\sigma: T_{\mathrm{SIG}}(X) \to T_{\mathrm{SIG}}(Y)$ is called a *substitution* if

$$\begin{aligned} \mathrm{sort}(\sigma(x)) &= \mathrm{sort}(x) && \text{for all variables } x\in X, \\ \sigma(\mathrm{op}\, t_1 \ldots t_k) &= \mathrm{op}\, \sigma(t_1) \ldots \sigma(t_k) && \text{for all other terms } \mathrm{op}\, t_1 \ldots t_k. \end{aligned}$$

TERM REWRITING. – A *rewrite rule* is a pair $l\to r$ of terms of equal sort such that l is not a variable and all variables in r occur also in l. A set $\mathscr{R}$ of rewrite rules is called a *term rewriting system.*

Given a rewriting system $\mathscr{R}$, we say that t *rewrites to* u, written $t\underset{\mathscr{R}}{\to} u$, if there is a rewrite rule $l\to r$ and a substitution σ such that t has a subterm $t'=\sigma(l)$, and u is obtained from t by replacing t' by $\sigma(r)$.

General assumption

For the rest of this paper, we fix a signature SIG and an infinite variable set X.

3. JUNGLES AND JUNGLE MORPHISMS

Representing terms such that common subterms can be shared requires graph-like structures. We use hypergraphs, an extension of graphs where hyperedges point from strings of source nodes to strings of target nodes.

3.1. Definition (hypergraph)

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 $m_G: E_G \to \mathrm{OP}$, labeling nodes with sorts, and hyperedges with operation symbols.

For a hypergraph G, $\mathrm{indegree}_G(v)$ and $\mathrm{outdegree}_G(v)$ denote the number of occurrences of a node v in the target strings resp. source strings of all edges in G.

Let v_1, v_2 be two nodes in a hypergraph G. Then $v_1 >_G v_2$ denotes that there is a non-empty path from v_1 to v_2 in G; $v_1 \geqq_G v_2$ means $v_1 >_G v_2$ or $v_1 = v_2$. G is *acyclic* if there is no node $v \in V_G$ such that $v >_G v$. □

Not all hypergraphs over SIG are suited to represent terms: The sources and targets of hyperedges must conform to the typing of SIG, each node should have at most one outgoing hyperedge (in order to represent a unique term), and cycles must not occur (in order to stay with finite terms).

3.2. Definition (jungle)

A hypergraph G is a *jungle* (*over* SIG) if

1. for each $e \in E_G$, $m_G(e) = \mathrm{op}: s_1 \ldots s_k \to s$ implies $l_G^*(s_G(e)) = s$ and $l_G^*(t_G(e)) = s_1 \ldots s_k$,
2. $\mathrm{outdegree}_G(v) \leqq 1$ for each $v \in V_G$,
3. G is acyclic. □

Remarks.

Generation. – In addition to the graph-theoretic definition above, the set of all jungles can be characterized by a set of generating jungle rules: each jungle can be constructed from the empty jungle by application of these rules. This characterization is used in [HKP88] to establish a structural induction principle for jungles.

Induction on nodes. – Since jungles are acyclic, two induction techniques are available for showing that all nodes of a given jungle have a property P: one shows that all nodes without ingoing edges fulfill P and that all target nodes of an edge satisfy P if the source node satisfies P. Alternatively, it is sufficient to show that all nodes without outgoing edges have property P and that the source node of an edge satisfies P if all the target nodes satisfy P. □

Nodes without outgoing hyperedges are places where a jungle is "incomplete". Thus these nodes are considered as variables.

3.3. Definition (variables in a jungle)

Let G be a jungle. Then $\mathrm{VAR}_G = \{ v \in V_G \mid \mathit{outdegree}_G(v) = 0 \}$ is the set of all *variables* in G.

In the following, we assume $\mathrm{VAR}_G \subseteq X$ for every jungle G. □

The term represented by some node in a jungle is obtained by descending along the hyperedges and collecting the hyperedge labels. This process terminates because jungles are acyclic, and yields a unique term because all nodes have at most one outgoing hyperedge.

3.4. Definition (term representation function)

Let G be a jungle. Then

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

defines a function $\mathrm{term}_G : V_G \to T_{\mathrm{SIG}}(\mathrm{VAR}_G)$ with $\mathrm{sort}(\mathrm{term}_G(v)) = l_G(v)$ for all $v \in V_G$.

The set $\mathrm{term}_G(V_G)$ of all terms represented by a jungle G is denoted by TERM_G. □

3.5. Example

Let SIG contain a sort nat and operation symbols

$$0 : \to \mathrm{nat} \qquad \mathrm{succ}, \mathrm{fib} : \mathrm{nat} \to \mathrm{nat} \qquad + : \mathrm{nat}\,\mathrm{nat} \to \mathrm{nat}$$

In figure 2 below we show three jungles representing the term $\mathrm{fib}(n) + \mathrm{fib}(n)$. (In examples we write terms with additional parentheses and use infix notation, for better readability.)

Nodes are drawn as circles (in most cases, we omit their labels; by default all nodes are assumed to be labelled by "nat"); hyperedges are drawn as boxes which are connected by lines with their unique source nodes, and by arrows with their target nodes (if there are any). The arrows are arranged from left to right in the order given by the target mapping. □

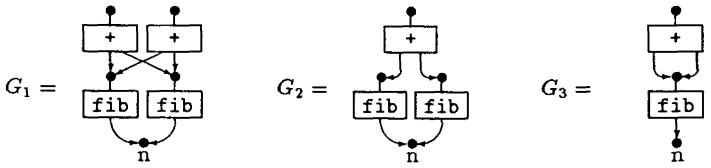


Figure 2. – Jungles representing fib (n) + fib (n).

It is easy to see that jungles are suited to represent arbitrary finite sets of terms.

3.6. Fact

For each finite set T of terms there is a jungle G with $T \subset TERM_G$. □

3.7. Definition (fully collapsed jungle)

A jungle G is called *fully collapsed* if $term_G$ is injective (*i. e.* if each term in TERM_G is represented by a unique node). □

For example, the jungle G_3 in figure 2 is fully collapsed.

Morphisms between jungles are essential for describing jungle manipulating rules and derivations; their role corresponds to that of substitutions for term rewriting.

3.8. Definition (jungle morphism)

Let G, H be jungles. A *jungle morphism* $f : G \to H$ is a pair of mappings $f = (f_V : V_G \to V_H,\ f_E : E_G \to E_H)$ which preserve sources, targets, and labels, *i. e.*, $s_H \circ f_E = f_V \circ s_G$, $t_H \circ f_E = f_V^* \circ t_G$, $l_H \circ f_V = l_G$, and $m_H \circ f_E = m_G$. f is called *injective* (*surjective*), if both f_V and f_E are injective (surjective). If f is injective and surjective, it is called an *isomorphism* and G and H are said to be *isomorphic*, written $G \cong H$. □

Due to the jungle structure, jungle morphisms are uniquely determined by their values for roots.

Notation. – $\mathrm{ROOT}_G = \{ v \in V_G \mid indegree_G(v) = 0 \}$ denotes the set of *roots* of a jungle G. □

3.9. Fact (uniqueness of jungle morphisms)

Let $f, g: G \to H$ be jungle morphisms. Then $f = g$ if and only if $f_V(r) = g_V(r)$ for all $r \in \mathrm{ROOT}_G$. □

Each jungle morphism $f: G \to H$ induces a substitution compatible with f.

3.10. Definition (induced substitution)

Let $f: G \to H$ be a jungle morphism. Then the *induced substitution*

$$\sigma: T_{\mathrm{SIG}}(\mathrm{VAR}_G) \to T_{\mathrm{SIG}}(\mathrm{VAR}_H)$$

is defined for all $x \in \mathrm{VAR}_G$ by

$$\sigma(x) = \mathrm{term}_H(f_V(x)). \quad \square$$

3.11. Fact

For each jungle morphism $f: G \to H$ the induced substitution σ satisfies

$$\sigma \circ \mathrm{term}_G = \mathrm{term}_H \circ f_V. \quad \square$$

Remark. – According to the above fact, the existence of a jungle morphism $G \to H$ implies that $\sigma(\mathrm{TERM}_G) \subseteqq \mathrm{TERM}_H$ for some substitution σ. The reverse does not hold as figure 2 shows: although

$$\mathrm{TERM}_{G_1} = \mathrm{TERM}_{G_2} = \mathrm{TERM}_{G_3},$$

there is no jungle morphism from G_3 to G_1 or to G_2. □

Let G and H be jungles where all roots of G represent terms that occur in H up to some substitution. Then a sufficient condition for the existence of a jungle morphism $G \to H$ is that H is fully collapsed.

3.12. Theorem (existence of jungle morphisms)

Let G, H be jungles and $f_{\mathrm{ROOT}}: \mathrm{ROOT}_G \to V_H$ be a mapping such that

$$\mathrm{term}_H(f_{\mathrm{ROOT}}(r)) = \sigma(\mathrm{term}_G(r)) \quad \textit{for all } r \in \mathrm{ROOT}_G$$

where $\sigma: T_{\mathrm{SIG}}(\mathrm{VAR}_G) \to T_{\mathrm{SIG}}(\mathrm{VAR}_H)$ is some substitution.

If H is fully collapsed, then f_{ROOT} uniquely extends to a jungle morphism $f: G \to H$ such that σ is the substitution induced by f.

Proof. – It is sufficient to show that f_{ROOT} can be extended to a jungle morphism with induced substitution σ. Uniqueness follows from fact. 3.9.

For each $v \in V_G$ there is some $r \in \mathrm{ROOT}_G$ such that $\mathrm{term}_G(v)$ is a subterm of $\mathrm{term}_G(r)$. Consequently, $\sigma(\mathrm{term}_G(v))$ is a subterm of $\mathrm{term}_H(f_{\mathrm{ROOT}}(r))$ and, by subterm-closedness of TERM_H, is represented by some $v' \in H_H$. v' is uniquely determined since H is fully collapsed. Hence the assignment $v \mapsto v'$ defines a mapping $f_V : V_G \to V_H$ which satisfies

$$\mathrm{term}_H(f_V(v)) = \sigma(term_G(v)) \quad \text{for all } v \in V_G. \tag{1}$$

We also have

$$l_H \circ f_V = l_G \tag{2}$$

as each node is labeled with the sort of its associated term.

Now consider some $e \in E_G$. Since $\mathrm{term}_G(s_G(e))$ is not a variable, $\mathrm{term}_H(f_V(s_G(e)))$ is not a variable, too. So there is (uniquely) $e' \in E_H$ with $s_H(e') = f_V(s_G(e))$. *I. e.*, the assignment $e \mapsto e'$ defines a mapping $f_E : V_G \to V_H$ with

$$s_H \circ f_E = f_V \circ s_G. \tag{3}$$

To complete the proof that (f_V, f_E) is a jungle morphism we must show

$$m_H \circ f_E = m_G \tag{4}$$

and

$$t_H \circ f_E = f_V^* \circ t_G. \tag{5}$$

For each $e \in E_G$ we have

$$\begin{aligned}
&m_H(f_E(e))\,\mathrm{term}_H^*(t_H(f_E(e))) \\
&\quad = \mathrm{term}_H(s_H(f_E(e))) && \text{definition 3.4} \\
&\quad = \mathrm{term}_H(f_V(s_G(e))) && (3) \\
&\quad = \sigma(\mathrm{term}_G(s_G(e))) && (1) \\
&\quad = \sigma(m_G(e)\,\mathrm{term}_G^*(t_G(e))) && \text{definition 3.4} \\
&\quad = m_G(e)\,\sigma^*(\mathrm{term}_G^*(t_G(e))) \\
&\quad = m_G(e)\,\mathrm{term}_H^*(f_V^*(t_G(e))) && (1)
\end{aligned}$$

which implies (4) and, by injectivity of term_H, (5). □

As a corollary we obtain that fully collapsed jungles which represent the same terms are isomorphic.

3.13. Corollary (uniqueness of fully collapsed jungles)

Let G, H be fully collapsed jungles. If $\mathrm{TERM}_G = \mathrm{TERM}_H$, *then G and H are isomorphic.*

Proof. – Let $g_{\mathrm{ROOT}} : \mathrm{ROOT}_G \to V_H$ and $h_{\mathrm{ROOT}} : \mathrm{ROOT}_H \to V_G$ map each root to a node representing the same term. By theorem 3.12, these mappings extend to jungle morphisms $g : G \to H$ and $h : H \to G$ such that $\mathrm{term}_G = \mathrm{term}_H \circ g_V$ and $\mathrm{term}_H = \mathrm{term}_G \circ h_V$. Hence $\mathrm{term}_G = \mathrm{term}_G \circ h_V \circ g_V$ which by injectivity of term_G implies $h_V \circ g_V = \mathrm{id}_{V_G}$ (where id_{V_G} is the identity on V_G). Analogously we get $g_V \circ h_V = \mathrm{id}_{V_H}$, so g_V and h_V are bijections. This implies injectivity of g_E and h_E while surjectivity follows from the fact that g_V and h_V preserve represented terms. Thus g and h are isomorphisms. □

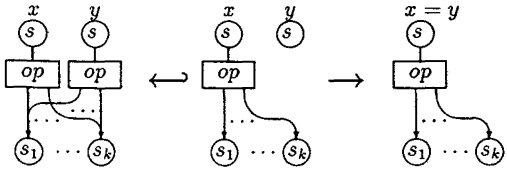
4. FOLDING

Given a jungle, we want to generate the fully collapsed jungle representing the same terms by application of *folding rules*. To explain the idea behind these rules, consider a jungle G and two hyperedges e_1, e_2 in G which have the same labels and target nodes but different sources. Then $s_G(e_1)$ and $s_G(e_2)$ represent the same term. Deleting one of the edges and identifying both source nodes yields a jungle which represents the same terms as G.

4.1. Definition (folding rule)

Let $\mathrm{op} : s_1 \ldots s_k \to s \in \mathrm{OP}$ be an operation symbol with $k \geqq 0$.

The *folding rule* for op is given by a pair $(L \hookleftarrow K \xrightarrow{b} R)$ of jungle morphisms as depicted below ("$x = y$" indicates that b identifies the roots of K; note that L and R have no variables if op is a constant).



$\mathscr{F}$ denotes the set of folding rules for OP. □

4.2. Definition (folding step)

Let G be a jungle. A *folding step* $G \underset{\mathscr{F}}{\Rightarrow} H$ from G to a hypergraph H is constructed as follows:

1. Find a morphism $g: L \to G$ for some folding rule $(L \supset K \xrightarrow{b} R)$ such that g_E is injective.

2. Obtain D from G by removing $g_E(e)$, where e is the unique edge in $L - K$.

3. Obtain H from D by identifying $g_V(x)$ and $g_V(y)$, where x and y are the roots in L.

We do not want to distinguish between H and hypergraphs that differ from H only in their names for edges or non-variables nodes. That is, we stipulate that $G \underset{\mathscr{F}}{\Rightarrow} H'$ whenever there is an isomorphism $f: H \to H'$ with $f_V(x) = x$ for all $x \in \mathrm{VAR}_H$. □

Remark. – The jungles involved in the construction of a folding step are related by morphisms as follows:

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

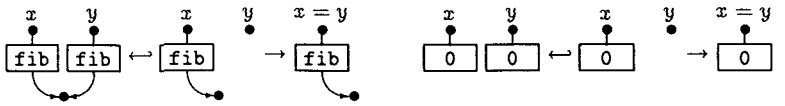
This diagram actually represents two pushouts in the category of hypergraphs and hypergraph morphisms (*see* [Ehr79] for a categorical approach to graph grammars). □

4.3. Definition (track function)

For each folding step $G \underset{\mathscr{F}}{\Rightarrow} H$ there is a *track function* $\mathrm{tr}: V_G \to V_H$ which coincides with c_V in the above diagram (note that $V_D = V_G$). □

4.4. Example

Below we show the folding rules for the operation symbol fib and the constant 0.



In example 3.5, the folding rule for fib transforms G_2 into G_3. □

4.5. Lemma (folding steps preserve jungles)

Given a jungle G and a folding step $G \underset{\mathcal{F}}{\Rightarrow} H$, *H is a jungle, too.*

Proof. – Let $G \underset{\mathcal{F}}{\Rightarrow} H$ be given with a diagram as shown in the remark below definition 4.1. H is obtained from D by identifying a node v with a variable x where v and x are the images of the two roots in K. Clearly H satisfies the outdegree and the labeling condition of definition 3.2. To show that H is acylic suppose that there is a cycle in H. Since D is acyclic this cycle is the result of identifying v and x, *i. e.*, there must be a path in D from v to x. So in particular $v >_G x$ because D is a subjungle of G. But by the structure of L, $v >_G v'$ implies $x >_G v'$ for all $v' \in V_G$. Thus $x >_G x$ which contradicts the acyclicity of G. □

4.6. Lemma (folding steps preserve terms)

Let $G \underset{\mathcal{F}}{\Rightarrow} H$ *be a folding step. Then* $\mathrm{term}_G = \mathrm{term}_H \circ \mathrm{tr}$ and $\mathrm{TERM}_G = \mathrm{TERM}_H$.

Proof. – It is easy to see that the construction of folding rules ensures that $\mathrm{term}_G(v) = \mathrm{term}_H(\mathrm{tr}(v))$ for all nodes v in G. Then $\mathrm{TERM}_G = \mathrm{TERM}_H$ follows from the observation that tr is surjective. □

4.7. Theorem (characterization of fully collapsed jungles)

A jungle G is fully collapsed if and only if there is no folding step $G \underset{\mathcal{F}}{\Rightarrow} H$.

Proof. – Let G be fully collapsed and L be the left-hand side of some folding rule. Then each jungle morphism $L \to G$ identifies the two edges in

L. This violates the condition that g_E must be injective, so there is no folding step $G \underset{\mathscr{F}}{\Rightarrow} H$.

Conversely, assume that G is not fully collapsed. Then there are $v_1, v_2 \in V_G$ with $v_1 \neq v_2$ but $\text{term}_G(v_1) = \text{term}_G(v_2)$. (Note that v_1 and v_2 cannot be variables.) In particular, we can choose v_1 and v_2 such that they are minimal with respect $>_G$, *i. e.*, for each pair (v_1', v_2') with $v_i >_G v_i' (i = 1, 2)$ we have that $\text{term}_G(v_1') = \text{term}_G(v_2')$ implies $v_1' = v_2'$. Then there are edges e_1, e_2 with $s_G(e_i) = v_i (i = 1, 2)$ and $t_G(e_1) = t_G(e_2)$. e_1 and e_2 are labeled with the same operation symbol op. Hence the folding rule for op can be applied. □

4.8. Theorem

$\underset{\mathscr{F}}{\Rightarrow}$ *is terminating and confluent.*

Proof. – Since each folding step decreases the number of nodes by one, folding is terminating. Thus, given derivations $H_1 \overset{*}{\underset{\mathscr{F}}{\Leftarrow}} G \overset{*}{\underset{\mathscr{F}}{\Rightarrow}} H_2$, we have derivations $H_i \overset{*}{\underset{\mathscr{F}}{\Rightarrow}} H_i'$ where H_i' is a normal form $(i = 1, 2)$. With theorem 4.7 and lemma 4.6 it follows that H_1' and H_2' are fully collapsed and represent the same terms. Hence $H_1' \cong H_2'$ by corollary 3.13. □

5. JUNGLE EVALUATION

In this section we study how term rewriting can be implemented by hypergraph replacement. We define rules which manipulate jungles such that the represented terms are rewritten according to a given term rewriting system.

GENERAL ASSUMPTION

Let $\mathscr{R}$ be an arbitrary but fixed term rewriting system over SIG.

5.1. Definition (evaluation rule)

The *evaluation rule* for a rewrite rule $l \to r$ is given by a pair $(L \supset K \overset{b}{\to} R)$ of jungle morphisms as follows:

1. L is a variable-collapsed tree with $\text{term}_L(\text{root}_L) = l$. (A jungle G is a *variable-collapsed tree* if it has a single root root_G, and if $\textit{indegree}_G(v) > 1$ implies $v \in \text{VAR}_G$, for all $v \in V_G$.)

2. K is the subjungle of L obtained by removing the edge with source root_L.

3. R is obtained from K as follows: if r is a variable, then the node root_L is identified with the node r; otherwise, R is the disjoint union of K and the variable-collapsed tree R' with $\text{term}_{R'}(\text{root}_{R'}) = r$ where $\text{root}_{R'}$ is identified with root_L and each $x \in \text{VAR}_{R'}$ is identified with its counterpart in VAR_L.

4. $K \subsetneq L$ and $K \overset{b}{\to} R$ are inclusions with the exception that b maps root_L to r if r is a variable. □

5.2. Definition (evaluation step)

Let G be a jungle and $p = (L \supset K \overset{b}{\to} R)$ the evaluation rule for $l \to r$.

An evaluation step $G \underset{p}{\Rightarrow} H$ via p is constructed as follows:

1. Find a morphism $g : L \to G$.

2. Remove the edge with source $g_V(\text{root}_L)$ from G to obtain a jungle D.

3. If r is a variable, identify the node $g_V(\text{root}_L)$ with the node $g_V(r)$ to get H; otherwise, construct the disjoint union $D + R'$ and identify $\text{root}_{R'}$ with $g_V(\text{root}_L)$, and each $x \in \text{VAR}_{R'}$ with $g_V(x)$ to obtain H.

As for folding steps, we stipulate that $G \underset{p}{\Rightarrow} H$ implies $G \underset{p}{\Rightarrow} H'$ if there is an isomorphism $f : H \to H'$ *with* $f_V(x) = x$ for all $x \in \text{VAR}_H$. □

Remark. – The jungles involved in the construction of evaluation steps are related by morphisms as in the case of folding steps (see the remark below definition 4.1). Since $V_D = V_G$ holds for evaluation steps as well, there is again a track function $\text{tr} : V_G \to V_H$. tr is injective on VAR_G so that we still may assume that tr does not rename variables. □

5.3. Example (evaluation rules and evaluation step)

In figure 3 we show two evaluation rules; note that the rule for $0+n \to n$ identifies the root m with the variable n.

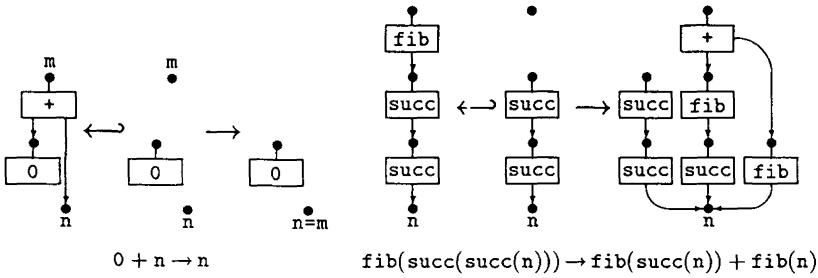


Figure 3. – Two evaluation rules.

Figure 4 shows an evaluation step via the left rule (the occurrence of the left-hand side is drawn with dashed boxes).

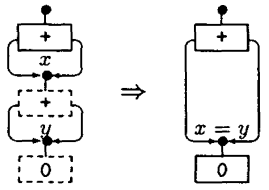


Figure 4. – Application of the evaluation rule $0+n \to n$.

The first two steps of figure 1 in the introduction show applications of the second evaluation rule of figure 3. □

5.4. Theorem (evaluation steps preserve jungles)

If G is a jungle and $G \underset{p}{\Rightarrow} H$ via some evaluation rule p, then H is a jungle too.

Proof. – Let p be the evaluation rule for $l \to r$, and let $g : L \to G$ be the morphism used to construct H.

Case 1: r is a variable. By definition 5.2, H is obtained from G by identifying $g_V(\text{root}_L)$ with some node $g_V(v)$. From $outdegree_D(g_V(\text{root}_L)) = 0$ we conclude that no node in H has an outdegree greater than one.

To show that H is acyclic suppose that there is a cycle in H. Then the construction of H implies $g_V(v) >_G g_V(\text{root}_L)$. However, $g_V(\text{root}_L) >_G g_V(v)$ holds because there is a path from root_L to every other node in L. Thus there is a cycle in G, contradicting the fact that G is a jungle.

Case 2 : r is not a variable. Then K is a subjungle of R. If p a *jungle rule* in the sense of [HKP88], then H is a jungle by the jungle preservation theorem presented in that paper. We have to show

(i) $\text{VAR}_L \subseteq \text{VAR}_R$ and

(ii) for each $v \in V_K$ and each $x \in \text{VAR}_L$, $v >_R x$ implies $v >_L x$.

(i) follows from the definition of p. Let therefore $v \in V_K$ and $x \in \text{VAR}_L$ with $v >_R x$. By definition of p, each new path in R between two nodes in K starts at root_L. So we only have to consider the case $\text{root}_L >_R x$. Since L is a variable-collapsed tree, we have $\text{root}_L >_L v$ for each $v \in V_L - \{\text{root}_L\}$ and in particular $\text{root}_L >_L x$. □

5.5. Evaluation theorem

Let $p = (L \supset K \overset{b}{\rightarrow} R)$ *be the evaluation rule for some rewrite rule* $l \rightarrow r$, *and let* $G \underset{p}{\Rightarrow} H$ *be an evaluation step under some morphism* $g : L \rightarrow G$.
Then for each $v \in V_G$,

(1) $\text{term}_G(v) \underset{\{l \rightarrow r\}}{\overset{n}{\rightarrow}} \text{term}_H(\text{tr}(v))$

where n is the number of paths from v *to* $g_V(root_L)$.

Proof. – Let the evaluation step have the diagram

$$\begin{array}{ccccc} L & \supset & K & \overset{b}{\rightarrow} & R \\ {\scriptstyle g}\downarrow & & \downarrow & & \downarrow{\scriptstyle h} \\ G & \supset & D & \overset{c}{\rightarrow} & H \end{array}$$

We proceed by bottom-up induction over V_G.

Induction Base. – Let x be a variable in G. By construction of p no edge is appended to x. Furthermore, x is identified at most with $g_V(\text{root}_L)$ where the edge outgoing from $g_V(\text{root}_L)$ is removed. So $\text{tr}(x)$ is a variable, and we get $\text{term}_G(x) = x = \text{term}_H(\text{tr}(x))$. Since there is no path from x to $g_V(\text{root}_L)$ we conclude that (1) holds with $n = 0$.

Induction Hypothesis: Let $e \in E_G$ such that all $v \in V_G$ with $s_G(e) >_G v$ satisfy (1).

Induction Step: We distinguish two cases.

Case 1: $s_G(e) = g_V(\text{root}_L)$. Let $x \in \text{VAR}_L$. Then $g_V(\text{root}_L) >_G g_V(x)$ because L is a variable-collapsed tree. Since G is acyclic, this implies that there is no path from $g_V(x)$ to $g_V(s_G(e))$. So $\text{term}_G(g_V(x)) = \text{term}_H(\text{tr}(g_V(x)))$ by induction hypothesis. For the substitutions σ_g and σ_h induced by g and h, respectively, this means that $\sigma_g(x) = \sigma_h(x)$ for all $x \in \text{VAR}_L$ since $\text{tr}(g_V(x)) = h_V(b_V(x))$ for all $x \in \text{VAR}_L$ (note that $\text{VAR}_L = \text{VAR}_R$).

Thus we obtain

$$\text{term}_G(g_V(\text{root}_L)) = \sigma_g(l) \underset{\{l \to r\}}{\to} \sigma_h(r) = \text{term}_H(\text{tr}(g_V(\text{root}_L))).$$

So $g_V(\text{root}_L)$ satisfies (1) with $n = 1$ since there is only one path from $g_V(\text{root}_L)$ to itself (*viz.* the empty path).

Case 2: $s_G(e) \neq g_V(\text{root}_L)$. Then $e \in E_D$. Let $e' = c_E(e)$. The morphism properties of c imply

(2) $\quad m_G(e) = m_H(e'), \qquad \text{tr}(s_G(e)) = s_H(e'), \qquad \text{and} \qquad \text{tr}^*(t_G(e)) = t_H(e').$

Let $t_G(e) = v_1 \ldots v_k$ for some $k \geqq 0$ and n_i be the number of paths from v_i to $g_V(\text{root}_L)$, for $i = 1 \ldots k$. Then

$$\begin{aligned}
\text{term}_G(s_G(e)) &= m_G(e)\, \text{term}_G(v_1) \ldots \text{term}_G(v_k) \\
&\overset{n_1}{\underset{\{l \to r\}}{\to}} m_G(e)\, \text{term}_H(\text{tr}(v_1)) \ldots \text{term}_G(v_k) \qquad \text{ind. hyp.} \\
&\quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
&\overset{n_k}{\underset{\{l \to r\}}{\to}} m_G(e)\, \text{term}_H(\text{tr}(v_1)) \ldots \text{term}_H(\text{tr}(v_k)) \qquad \text{ind. hyp.} \\
&= m_H(e')\, \text{term}^*_H(t_H(e')) \qquad (2) \\
&= \text{term}_H(s_H(e')) \\
&= \text{term}_H(\text{tr}(s_G(e))). \qquad (2)
\end{aligned}$$

Thus (1) holds because there are $n = \sum_{k}^{i=1} n_i$ paths from $s_G(e)$ to $g_V(\text{root}_L)$. □

5.6. Example (term rewriting performed by evaluation steps)

In the evaluation step of figure 4, the terms represented by the roots of the jungles are related by a two-step rewriting $(0+0)+(0+0) \xrightarrow[\mathscr{R}]{2} 0+0$. □

6. COMPUTING TERM NORMAL FORMS

This section is about the *completeness* of jungle evaluation. We investigate the following question: given a term t, can we compute a normal form of t by jungle evaluation (provided there exists one)?

In the first place, we consider the applicability of evaluation rules.

6.1. Example (non-left-linear rewrite rules)

Figure 5 shows a tree T and a fully collapsed jungle G, both representing $f(0, 0)$, as well as the evaluation rule for the rewrite rule $f(n, n) \to 0$ (where n is a variable).

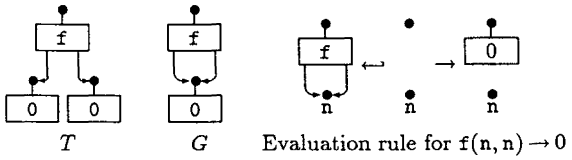


Figure 5. – Application of non-left-linear rules.

Although $f(0, 0)$ can be rewritten with $f(n, n) \to 0$, the evaluation rule cannot be applied to T: there is no jungle morphism from the left-hand side to T. This problem arises only with *non-left-linear* rewrite rules, *i.e.* rewrite rules which contain more than one occurrence of some variable on their left-hand side.

Theorem 3.12 suggests the following solution to this problem: prior to evaluation steps, jungles are transformed into fully collapsed jungles by application of folding steps. In this example, one application of the folding rule for 0 transforms T into G so that the evaluation rule can be applied subsequently.

6.2. Lemma (applicability of evaluation rules)

Let $p=(L \hookleftarrow K \overset{b}{\rightarrow} R)$ *be an evaluation rule for some rewrite rule* $l \rightarrow r$, *and* G *be a jungle such that* $l \rightarrow r$ *is applicable to some term* t *in* $TERM_G$. *Then there are jungles* $\bar{G}$ *and* H *such that* $G \overset{*}{\underset{\mathscr{F}}{\Rightarrow}} \bar{G} \underset{p}{\Rightarrow} H$.

Proof. – By theorems 4.7 and 4.8 there is a sequence of folding steps $G \overset{*}{\underset{\mathscr{F}}{\Rightarrow}} \bar{G}$ such that $\bar{G}$ is fully collapsed. Since $\mathrm{TERM}_G = \mathrm{TERM}_{\bar{G}}$ there is a node v *in* $\bar{G}$ such that $\mathrm{term}_{\bar{G}}(v) = \sigma(l)$ for some substitution σ. Hence, by theorem 3.12 there is a jungle morphism $g : L \rightarrow \bar{G}$ which maps root_L to v. So there is an evaluation step $\bar{G} \underset{p}{\Rightarrow} H$. □

Notation. – $\mathscr{G}(\mathscr{R})$ denotes the set of all evaluation and folding rules for $\mathscr{R}$. A *reduction step* $G \underset{\mathscr{G}(\mathscr{R})}{\Rightarrow} H$ is either an evaluation or a folding step. A sequence of reduction steps $G \overset{*}{\underset{\mathscr{G}(\mathscr{R})}{\Rightarrow}} H$ is called a *reduction* and $\mathrm{tr} : V_G \rightarrow V_H$ is the composition of the track functions of all reduction steps in this sequence. □

6.3. Normal form theorem

A jungle G *is a normal form with respect to* $\underset{\mathscr{G}(\mathscr{R})}{\Rightarrow}$ *if and only if all terms in* $TERM_G$ *are normal forms and* G *is fully collapsed.*

Proof. – Let G be fully collapsed and all terms in TERM_G be normal forms and suppose that there is a reduction step $G \underset{\mathscr{G}(\mathscr{R})}{\Rightarrow} H$. If this is an evaluation step, the evaluation theorem 5.5 yields the contradiction that $t \underset{\mathscr{R}}{\rightarrow} t'$ for some $t \in \mathrm{TERM}_G$ and some term t'. Otherwise, if this is a folding step, theorem 4.7 leads to the contradiction that G is not fully collapsed.

Conversely, let G be a normal form. Then G is fully collapsed by theorem 4.7. Moreover, if some $t \in \mathrm{TERM}_G$ would not be a normal form, then lemma 6.2 would lead to the contradiction that $G \overset{*}{\underset{\mathscr{F}}{\Rightarrow}} \bar{G} \underset{\mathscr{E}}{\Rightarrow} H$. □

6.4. Corollary

Let $G \overset{*}{\underset{\mathcal{G}(\mathcal{R})}{\Rightarrow}} H$ *be a reduction such that* H *is a normal form. Then for each* $v \in V_G$, $term_H(\mathrm{tr}(v))$ *is a normal form of* $term_G(v)$.

Proof. – By the evaluation theorem 5.5 and lemma 4.6 we have $\mathrm{term}_G(v) \overset{*}{\underset{\mathcal{R}}{\to}} \mathrm{term}_H(\mathrm{tr}(v))$ for each $v \in V_G$. Theorem 6.3 shows that $\mathrm{term}_H(\mathrm{tr}(v))$ is a normal form. □

6.5. Example (incompleteness of jungle evaluation)

Assume that $\mathcal{R}$ consists of the rewrite rules

$$f(x) \to g(x, x) \quad c \to d \quad d \to c \quad g(c, d) \to e$$

where c, d, and e are constant symbols, and x is a variable.

Then $f(c)$ has a normal form obtained by $f(c) \underset{\mathcal{R}}{\to} g(c, c) \underset{\mathcal{R}}{\to} \mathrm{g(c, d)} \underset{\mathcal{R}}{\to} e$. This rewrite sequence cannot be simulated by a jungle reduction sequence because the application of the evaluation rule for $f(x) \to g(x, x)$ to a jungle representing $f(c)$ yields a jungle which represents $g(c, c)$ such that the two occurrences of c are shared. Hence these occurrences cannot be evaluated independently and the evaluation rule for $g(c, d) \to e$ can never be applied. As a consequence, no jungle representing $f(c)$ has a normal form.

For similar reasons it may happen that jungle evaluation terminates but fails to compute all term normal forms. Consider the rewrite system

$$f(x) \to g(x, x) \quad c \to d \quad c \to e$$

Jungle evaluation can compute only $g(d, d)$ or $g(e, e)$ as normal forms of $f(c)$ although $g(d, e)$ and $g(e, d)$ are further normal forms. □

7. TERMINATION

In this section we prove that $\underset{\mathcal{G}(\mathcal{R})}{\Rightarrow}$ is terminating provided that the underlying rewrite system $\mathcal{R}$ is terminating. The proof is somewhat involved since in contrast to term rewriting, jungle evaluation steps may produce "garbage" which gives rise to additional evaluation steps.

7.1. Termination theorem

If $\underset{\mathscr{R}}{\rightarrow}$ *is terminating, then* $\underset{\mathscr{G}(\mathscr{R})}{\Rightarrow}$ *is terminating, too.*

Proof. – Let $\underset{\mathscr{R}}{\rightarrow}$ be terminating. In a first step we extend $\underset{\mathscr{R}}{\overset{+}{\rightarrow}}$ to the relation $>$ by defining

$$t > u \text{ if } u \text{ is a subterm of some } t' \text{ with } t \underset{\mathscr{R}}{\overset{+}{\rightarrow}} t'.$$

Like $\underset{\mathscr{R}}{\overset{+}{\rightarrow}}$, $>$ is a strict ordering. Moreover, $>$ is terminating as for each sequence $t_1 > t_2 > t_3 > \ldots$ there is a sequence $\hat{t}_1 \underset{\mathscr{R}}{\overset{+}{\rightarrow}} \hat{t}_2 \underset{\mathscr{R}}{\overset{+}{\rightarrow}} \hat{t}_3 \underset{\mathscr{R}}{\overset{+}{\rightarrow}} \ldots$ with t_i being a subterm of $\hat{t}_i$. In a second step $>$ is extended to an ordering $\gg$ on finite multisets of terms. (The multisets we consider are functions $\mathsf{M} : T_{\mathrm{SIG}}(X) \rightarrow \mathcal{N}$ with $\mathsf{M}(t) = 0$ almost everywhere. Then $\mathsf{M} \cup \mathsf{N}(t) = \mathsf{M}(t) + \mathsf{N}(t)$ and $\mathsf{M} - \mathsf{N}(t) = \max(0, \mathsf{M}(t) - \mathsf{N}(t))$ for all multisets M, N.) Following Dershowitz and Manna [DM79], we define $\gg$ by

$$\mathsf{A} \gg \mathsf{B} \text{ if } \mathsf{B} = (\mathsf{A} - \mathsf{REM}) \cup \mathsf{ADD}$$

where

- $\varnothing \neq \mathsf{REM} \subseteq \mathsf{A}$, and
- for each $u \in \mathsf{ADD}$ there is $t \in \mathsf{REM}$ with $t > u$.

In [DM79] it is shown that the termination of $>$ carries over to $\gg$.

We assign a multiset TERM_G to each jungle G by

$$\mathsf{TERM}_G(t) = |\{ v \in V_G \mid \mathrm{term}_G(v) = t \}|.$$

It remains to be shown that for all jungles G, H,

$$G \underset{\mathscr{G}(\mathscr{R})}{\Rightarrow} H \text{ implies } \mathsf{TERM}_G \gg \mathsf{TERM}_H.$$

For $G \underset{\mathscr{F}}{\Rightarrow} H$ we have $\mathsf{TERM}_H = \mathsf{TERM}_G - \{ t \}$ where t is the term represented by the two nodes in G which are identified. Let therefore $G \underset{p}{\Rightarrow} H$ for some evaluation rule $p = (L \supseteq K \rightarrow R)$ and let v be the image of root_L in G. Then $\mathsf{TERM}_H = (\mathsf{TERM}_G - \mathsf{REM}) \cup \mathsf{ADD}$ where

$$\mathsf{REM}(t) = |\{ \hat{v} \in V_G \mid \hat{v} \geqq_G v \text{ and } \mathrm{term}_G(\hat{v}) = t \}|$$

and

$$\mathsf{ADD} = \mathsf{OLD} \cup \mathsf{NEW}$$

with

$$\mathsf{OLD}(t) = |\{o \in V_H \mid o \geqq_H \mathrm{tr}(v) \text{ and } \mathrm{term}_H(o) = t\}|$$

and

$$\mathsf{NEW}(t) = |\{n \in V_H \mid n \notin \mathrm{tr}(V_G) \text{ and } \mathrm{term}_H(n) = t\}|.$$

Clearly we have $\mathsf{REM} \neq \varnothing$ because $\mathrm{term}_G(v) \in \mathsf{REM}$. Now consider some $u \in \mathsf{ADD}$.

Case 1: $u \in \mathsf{OLD}$. Then there are $\hat{v} \in V_G$ with $\hat{v} \geqq_G v$ and $o \in V_H$ such that $\mathrm{tr}(\hat{v}) = o$ and $\mathrm{term}_H(o) = u$. By theorem 5.5 we have $\mathrm{term}_G(\hat{v}) \overset{+}{\underset{\mathscr{R}}{\to}} u$, so $\mathsf{REM} \ni \mathrm{term}_G(\hat{v}) > u$.

Case 2: $u \in \mathsf{NEW}$. Then there is $n \in V_H - \mathrm{tr}(V_G)$ with $\mathrm{term}_H(n) = u$. $n \notin \mathrm{tr}(V_G)$ implies $\mathrm{tr}(v) >_H n$, *i.e.*, u is a subterm of $\mathrm{term}_H(\mathrm{tr}(v))$. We conclude $\mathsf{REM} \ni \mathrm{term}_G(v) > u$ since $\mathrm{term}_G(v) \overset{+}{\underset{\mathscr{R}}{\to}} \mathrm{term}_H(\mathrm{tr}(v))$ (by theorem 5.5).

Thus $\mathsf{TERM}_G \gg \mathsf{TERM}_H$ which shows that $\underset{\mathscr{G}(\mathscr{R})}{\Rightarrow}$ is terminating. □

Theorem 7.1 together with corollary 6.4 suggests that jungle evaluation is particularly suited for the implementation of terminating and confluent term rewriting systems. Each term represented by some jungle can be evaluated through the exhaustive application of evaluation and folding rules. For the implementation one may choose any evaluation strategy since every jungle reduction computes the unique normal form of the input term.

8. CONFLUENCE

In this section we show by a counterexample that confluence does not carry over from term rewriting to jungle evaluation. It turns out, though, that confluence is preserved if the given term rewriting system is also terminating, and if the "garbage" produced by evaluation steps is ignored.

8.1. Example

Consider the following rewrite system:

Sorts s, t **Operations**	$c : \to s$	**Rules**	$c \to d$
	$d : \to s$		$d \to c$
	$e : \to t$		$f(c) \to e$
	$f : s \to t$		$f(x) \to g(x, x)$
	$g : ss \to t$		$g(c, d) \to e$

Figure 6 demonstrates that the rewrite relation of this system is confluent while jungle evaluation is not confluent.

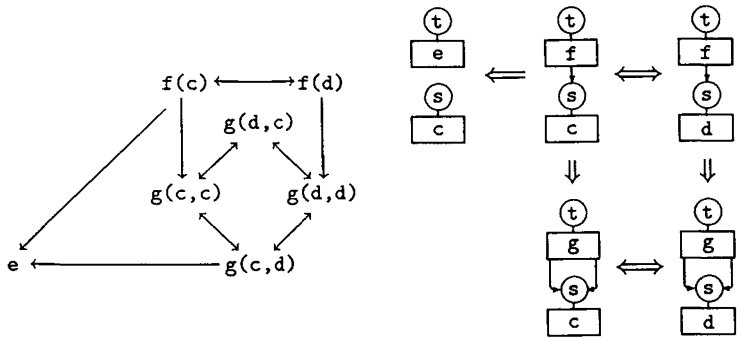


Figure 6. – Non-confluence caused by sharing.

The rewriting step $g(c, c) \to g(c, d)$ cannot be simulated since the evaluation step corresponding to $f(c) \to g(c, c)$ generates a jungle in which both occurrences of c are represented by the same node. Analogously, neither $g(d, d) \to g(c, d)$ can be simulated. □

The above rewriting system is neither terminating nor non-overlapping. It turns out that these conditions are essential for the confluence of jungle evaluation. (*See* the long version of [HP88] for the case of non-overlapping term rewrite systems.) Below we consider the case that $\underset{\mathscr{R}}{\to}$ is terminating. For achieving confluence we have to ignore the "garbage" produced by evaluation rules.

8.2. Example (non-confluence caused by garbage)

Let $c \to d$ and $f(c) \to f(d)$ be rewrite rules for an operation symbol f and constant symbols c and d. Figure 7 demonstrates that term rewriting is confluent, whereas jungle reduction yields distinct normal forms.

This is because the argument c is removed when the rewrite rule $f(c) \to f(d)$ is applied, while the corresponding subjungle (drawn with dashed box and unfilled circle) is preserved by the evaluation step. This subjungle is *garbage* with respect to the term rewriting performed at the root. □

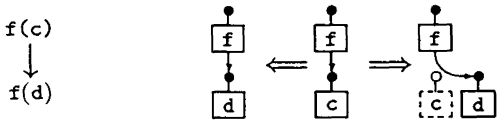


Figure 7. – Non-confluence caused by garbage.

In the following we distinguish certain nodes in a jungles as "points of interest", and consider the nodes and edges that are not reachable from these points as *garbage*. It is possible to remove garbage from jungles explicitly by *garbage collecting rules* (which are the inverses of the jungle generating rules defined in [HKP88]). Here we just restrict our attention to reductions that keep track of these points.

8.3. Definition (pointed jungle and pointed reduction)

A *pointed jungle* is a pair $(G\ ,\ P_G)$ where G is a jungle and $P_G \subseteq V_G$. We denote a pointed jungle only by its jungle component. $G^\bullet$ is the subjungle of G with $V_{G^\bullet} = \{ v' \in V_G \mid v \geqq_G v' \text{ for some } v \in P_G \}$ and

$$E_{G^\bullet} = \{ e \in E_G \mid s_G(e),\ t_G(e) \in V_{G^\bullet}^* \}.$$

A reduction $G \overset{*}{\underset{\mathcal{G}(\mathcal{R})}{\Rightarrow}} H$ is called *pointed* if $P_H = \mathrm{tr}(P_G)$. □

8.4. Theorem (confluence of pointed reduction)

Let $\underset{\mathcal{R}}{\to}$ *be terminating and confluent and let* $H_1 \overset{*}{\underset{\mathcal{G}(\mathcal{R})}{\Leftarrow}} G \overset{*}{\underset{\mathcal{G}(\mathcal{R})}{\Rightarrow}} H_2$ *be pointed reductions. Then there are pointed reductions* $H_1 \overset{*}{\underset{\mathcal{G}(\mathcal{R})}{\Rightarrow}} X_1$ *and* $H_2 \overset{*}{\underset{\mathcal{G}(\mathcal{R})}{\Rightarrow}} X_2$ *such that* $X_1^\bullet \cong X_2^\bullet$.

Proof. – By the termination theorem 7.1 there are pointed reductions

$H_i \overset{*}{\underset{\mathcal{G}(\mathcal{R})}{\Rightarrow}} X_i$ such that X_i (and thus $X_i^\bullet$) are normal forms for $i=1, 2$. We show that $\mathrm{TERM}_{X_1} = \mathrm{TERM}_{X_2}$. "$\subseteq$": Let v be a node in $X_1^\bullet$. Then there is a point p_1 in X_1 with $p_1 \geqq_{X_1} v$, so $\mathrm{term}_{X_1}(v)$ is a subterm of $\mathrm{term}_{X_1}(p_1)$.

Let tr_1 be the track function of $G \overset{*}{\underset{\mathcal{G}(\mathcal{R})}{\Rightarrow}} H_1 \overset{*}{\underset{\mathcal{G}(\mathcal{R})}{\Rightarrow}} X_1$ and let p be a point in G with $\mathrm{tr}_1(p) = p_1$. Now consider $p_2 = \mathrm{tr}_2(p)$, where tr_2 is the track function of $G \overset{*}{\underset{\mathcal{G}(\mathcal{R})}{\Rightarrow}} H_2 \overset{*}{\underset{\mathcal{G}(\mathcal{R})}{\Rightarrow}} X_2$. By corollary 6.4 both $\mathrm{term}_{X_1}(p_1)$ and $\mathrm{term}_{X_2}(p_2)$ are normal forms of $\mathrm{term}_G(p)$. Since $\underset{\mathcal{R}}{\rightarrow}$ is confluent we conclude that $\mathrm{term}_{X_1}(p_1) = \mathrm{term}_{X_2}(p_2)$.

So $\mathrm{term}_{X_1}(v)$ is a subterm of $\mathrm{term}_{X_2}(p_2)$; hence there is a node v' in X_2 with $p_2 \geqq_{X_2} v'$ and $\mathrm{term}_{X_2}(v') = \mathrm{term}_{X_1}(v)$. Since v' belongs to $X_2^\bullet$ we get

$$\mathrm{term}_{X_1}(v) = \mathrm{term}_{X_2}(v') \in \mathrm{TERM}_{X_2}.$$

(Note that $\mathrm{term}_H(u) = \mathrm{term}_{H^\bullet}(u)$ for each pointed jungle H and each node u in $H^\bullet$.)

"$\supseteq$": By a symmetric argument.

Thus $\mathrm{TERM}_{X_1} = \mathrm{TERM}_{X_2}$. Moreover, X_1 and X_2 are fully collapsed by theorem 4.7, so the uniqueness theorem 3.13 yields $X_1^\bullet \cong X_2^\bullet$. □

9. SUMMARY AND OUTLOOK

We have shown how term rewriting can be (abstractly) implemented by graph rewriting within the jungle approach set up by [Plu86] and [HKP88].

Jungles are acyclic hypergraphs which represent finite sets of terms such that common subterms can be shared. In particular, fully collapsed jungles represent terms such that all equal subterms are shared. Every jungle can be transformed into a unique fully collapsed jungle by applying folding rules. Term rewrite rules are translated into evaluation rules which are sound in the sense that each evaluation step performs a parallel term rewriting step. When evaluation and folding rules are used together, jungle normal forms always represent term normal forms, even for non-left-linear term rewriting systems. Furthermore, terminating term rewriting systems result in terminating jungle systems. Although confluence is not preserved in general, terminating and confluent term rewriting systems result in confluent jungle evaluation systems if the garbage produced by evaluation steps is ignored.

Comparaison with the approach of Barendregt *et al.*

In their paper [BvEG*87], Barendregt, van Eekelen, Glauert, Kennaway, Plasmeijer, and Sleep have proposed an alternative graph rewriting approach which is specially designed to model term rewriting.

While [BvEG*87] mainly focusses on strategies to compute term normal forms, we do not consider strategies in this paper. (See [Plu91] for results on jungle evaluation strategies). Instead, we consider termination and confluence of jungle evaluation, depending on properties of the given term rewriting system.

The main completeness result of [BvEG*87] states that term graphs representing normal forms can be computed for weakly regular term rewrite systems (where "weakly regular" essentially means left-linear and non-overlapping). We have shown that jungle reduction is complete for arbitrary terminating and confluent term rewriting systems.

The use of folding rules in our approach allows us to consider arbitrary term rewriting systems. [BvEG*87] treats only left-linear systems because graph rewriting without folding is incomplete otherwise. Moreover, folding does not only guarantee completeness but also speeds up the evaluation: the change from exponential time to linear time in the Fibonacci example shown in the introduction is only possible with folding.

Outlook

The work presented here should be continued in various directions:

Complexity analysis: In general, the space required for term rewriting grows exponentially with the number of rewrite steps since the size of a term may be multiplied in a single step. In contrast, jungle evaluation needs only linear space because each evaluation step increases the size of a jungle only by a constant.

Beside this general observation, the time and space complexity of term rewriting and jungle evaluation should be compared in a precise way. Classes of term rewrite systems should be identified for which jungle evaluation is strictly more efficient than term rewriting.

Parallel evaluation: Jungle evaluation is based on the algebraic approach of graph grammars for which parallelism and concurrency concepts have been developed (*see*, e. g., [Ehr83], [KW87]). It should be possible to exploit results from these areas to work out a theory of parallel jungle evaluation.

General jungle manipulation: The evaluation and folding rules used in this paper are special cases of hypergraph rules which preserve the jungle structure.

Such *jungle rules* have been introduced in [Plu86] and [HKP88] as a general means for deriving jungles from jungles. One could think of a specification and programming approach based on these rules which goes beyond the possibilities of term rewriting.

ACKNOWLEDGEMENT

We thank the anonymous referee for helpful comments.

REFERENCES

[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, *Proc. PARLE, Lecture Notes in Comp. Sci.*, 1987, *259*, pp. 141-158.

[DJ90] N. Dershowitz and J.-P. Jouannaud, Rewrite Systems, *Handbook of Theoretical Computer Science*, Vol. B, chapter 15, North Holland, 1990.

[DM79] N. Dershowitz and Z. Manna, Proving Termination with Multiset Orderings, *Comm. ACM*, 1979, *22*, (8), pp. 465-476.

[Ehr79] H. Ehrig, Introduction to the Algebraic Theory of Graph Grammars, *Proc. 1st Graph Grammar Workshop, Lecture Notes in Comp. Sci.*, 1979, *73*, pp. 1-69.

[Ehr83] H. Ehrig, Aspects of Concurrency in Graph Grammars, *Proc. 2nd Graph Grammar Workshop, Lecture Notes in Comp. Sci.*, 1983, *153*, pp. 58-81.

[ER76] H. Ehrig and B. K. Rosen, *Commutativity of Independent Transformations on Complex Objects*, Research Report RC 6251, IBM T. J. Watson Research Center, Yorktown Heights, 1976.

[HKP88] A. Habel, H.-J. Kreowski and D. Plump, Jungle Evaluation, *Proc. Fifth Workshop on Specification of Abstract Data Types. Lecture Notes in Comput. Sci.*, 1988, *332*, pp. 92-112. Revised version to appear in *Fundamentae Informaticae*.

[Hof83] B. Hoffmann, Compiler Generation: From Language Descriptions to Abstract Compilers, *Dissertation*, TU Berlin, 1983.

[HP88] B. Hoffmann and D. Plump, Jungle Evaluation for Efficient Term Rewriting, *Proc. Algebraic and Logic Programming, Lecture Notes in Comput. Sci.*, 1988, *343*, pp. 191-203.

[Klo90] J. W. Klop, Term Rewriting Systems: from Church-Rosser to Knuth-Bendix and Beyond, *Proc. ICALP'90, Lecture Notes in Comput. Sci.*, 1990, *443*, pp. 350-369.

[KW87] H.-J. Kreowski and A. Wilharm, Is Parallelism Already Concurrency? – Part 2: Non-sequential Processes in Graph Grammars, *Proc. 3rd Graph Grammar Workshop, Lecture Notes in Comput. Sci.*, 1987, *291*, pp. 361-377.

[Pad82] P. Padawitz, Graph Grammars and Operational Semantics, *Theoret. Comput. Sci.*, 1982, *19*, pp. 117-141.

[Plu86] D. Plump, Im Dschungel: Ein neuer Graph-Grammatik-Ansatz zur effizienten Auswertung rekursiv definierter Funktionen, *Diplomarbeit*, Fachbereich Mathematik/Informatik, Universität Bremen, 1986.

[Plu91] D. Plump, Graph-Reducible Term Rewriting Systems, *Proc. 4th Graph Grammar Workshop, Lecture Notes in Comput. Sci.*, 1991 (to appear).

[Rao84] J. C. Raoult, On Graph Rewritings, *Theoret. Comput. Sci.*, 32, 1984, pp. 1-24.

[Sta80] J. Staples, Computations on Graph-like Expressions, *Theoret. Comput. Sci.*, 1980, *10*, pp. 171-185.