Term Rewriting as Interaction Nets Plus Hyperedge Replacement

Markus Lohrey

Grundlagen der Programmierung, Institut für Softwaretechnik I,
Fakultät Informatik, Technische Universität Dresden,
D–01062 Dresden, Germany,
e–mail: lohrey@orchid.inf.tu-dresden.de

Abstract. In this paper we show that an arbitrary term rewriting system can be simulated by a set of hypergraph rewrite rules that can be split into two parts. The first part is a set of hyperedge replacement rules, one for each term rewrite rule. The second part is equivalent to an interaction net.

1 Introduction

The use of graphs for the efficient implementation of *term rewriting* is widely known. The advantage in the use of graphs is that identical subterms of a term may be shared and thus have only to be represented once. In the last decades, a great number of different approaches exploiting this basic idea have been developed, see [SPvE93] for an overview.

Roughly speaking, the great majority of these approaches operates in the following way. If $s \to t$ is a *term rewrite rule*, a corresponding *graph rewrite rule* $S \to T$ is constructed, where the graph S (T) represents s (t). If this rule is applied to a graph G, which represents another term, an occurrence of S inside G is replaced by T. Since the surroundings of the occurrence of S are not affected by this operation, this approach acts locally. On the other hand, the occurrence of S in G that has to be replaced may be arbitrarily large. Thus, the amount of work for checking applicability may be arbitrarily large. [HP91] is an example of this approach, which uses special term–representing *hypergraphs* (see Section 2.2 for the definition of a hypergraph), so called *jungles*.

A completely different approach was developed in [CR93] under the name *hyperedge replacement jungle rewriting*. In this approach, a term rewrite rule can be represented by a single *hyperedge replacement rule*, i.e., a hypergraph rewrite rule that replaces only a single hyperedge. Thus, every step in this formalism needs the same amount of work for checking applicability. The disadvantage of [CR93] is that the application of a single hyperedge replacement jungle rewrite rule may modify the whole hypergraph and therefore does not act locally.

In this paper, we will integrate the advantages of [HP91] and [CR93]. For an arbitrary term rewriting system $\mathcal{R}$, we will construct a hypergraph rewriting system $\mathcal{H}(\mathcal{R})$ that simulates $\mathcal{R}$ in a sound and complete way. The rules of $\mathcal{H}(\mathcal{R})$ act locally and replace only a small part of the whole hypergraph, namely at most two hyperedges. Moreover, $\mathcal{H}(\mathcal{R})$ (or more accurately, a subset of $\mathcal{H}(\mathcal{R})$ that is still sound and complete but less efficient) splits into two parts. The first part is essentially the set of hyperedge replacement rules used in [CR93]. The only difference concerns the sharing and discarding of terms, which has to be made explicitly in our approach by special duplicator edges and garbage collection edges, which appear in several graph reduction formalisms, e.g., [Lam90], [Laf90], [Laf95]. The second part of $\mathcal{H}(\mathcal{R})$ does not depend on $\mathcal{R}$ and is equivalent to an *interaction net*, see [Laf90]. Interaction nets are a generalization of proof nets in linear logic. They are a local and deterministic model of computation, where confluence is guaranteed. The encoding of a restricted class of term rewriting systems into interaction nets was also investigated in [FM96a], see Section 5 for a comparison with our work.

The paper is organized as follows. Section 2 introduces the necessary preliminaries. In Section 3.1, we construct the hypergraph rewrite system $\mathcal{H}(\mathcal{R})$ mentioned above, whose soundness (correctness) is proved in Section 3.2 (3.3). In Section 4 we present the splitting of $\mathcal{H}(\mathcal{R})$ into a hyperedge replacement system and an interaction net. Finally, in Section 5 we compare our approach with other approaches and discuss several further possible applications.

2 Preliminaries

2.1 Basic notations

Let A be a set. If A is finite, $|A|$ denotes the number of elements in A. A^* denotes the set of all finite lists of elements of A, including the empty list, which will be denoted by ϵ. Underlined letters will always denote lists. Let $\underline{w} \in A^*$. $\underline{w}_i$ denotes the ith element in $\underline{w}$ (starting with $\underline{w}_1$). $|\underline{w}|$ denotes the length of $\underline{w}$. If a list $\underline{w}$ appears at a position, where a set would be expected (e.g., $x \in \underline{w}$ or $\underline{w} \cup \underline{v}$) then $\underline{w}$ denotes the set of its members. Let $f : A \to B$ and $g : B \to C$ be (partial) functions. The composition $f \circ g$ is defined by $f \circ g(x) = f(g(x))$. $f^* : A^* \to B^*$ is the function defined inductively by (i) $f(\epsilon) = \epsilon$ and (ii) $f^*(wa) = f^*(w)f(a)$ for $w \in A^*$ and $a \in A$. The *transitive closure* of a relation $\to \subseteq A \times A$ is denoted by $\to^+$. The *transitive and reflexive closure* of $\to$ is denoted by $\to^*$.

For the rest of this paper we fix an alphabet Σ of *function symbols* as well as a countable infinite set Var of *variables*. $\Sigma^{(n)}$ denotes the set of all n–placed function symbols in Σ. $T_\Sigma(Var)$ denotes the set of all *terms* built using function symbols from Σ and variables from Var. $T_\Sigma(\emptyset)$ is abbreviated by T_Σ. $Var(t)$ denotes the set of all variables that appear in $t \in T_\Sigma(Var)$. $\mathcal{O}(t) \subseteq \mathbb{N}^*$, the set of all *occurrences of* t, is defined inductively by (i) $\epsilon \in \mathcal{O}(t)$ and (ii) $\mathcal{O}(f(t_1, \ldots, t_n)) = \{io \mid 1 \leq i \leq n \wedge o \in \mathcal{O}(t_i)\}$. Let $o \in \mathcal{O}(t)$. The *subterm of* t *at occurrence* o, denoted by $t[o]$ is inductively defined by (i) $t[\epsilon] = t$ and (ii) $f(t_1, \ldots, t_n)[io'] = t_i[o']$. $t[o \leftarrow s]$ is the result of replacing $t[o]$ in t by s. A *substitution* is a function $\theta : Var \to T_\Sigma(Var)$, such that the set $dom(\theta) = \{x \in Var \mid \theta(x) \neq x\}$ is finite. The unique homomorphic extension of θ to $T_\Sigma(Var)$ will be denoted by θ as well. We also use the notation $t\{\theta(x)/x \mid x \in dom(\theta)\} = \theta(t)$. θ is called *ground* if $\theta(x) \in T_\Sigma$ for every $x \in dom(\theta)$.

A *term rewriting system* is a finite set $\mathcal{R}$ of pairs (s', t'), where $s', t' \in T_\Sigma(Var)$, $s' \notin Var$ and $Var(t') \subseteq Var(s')$. Instead of (s', t') we use the more common notation $(s' \rightarrow t')$. If $s, t \in T_\Sigma$, we write $s \rightarrow_{\mathcal{R}} t$ iff there exist a $(s \rightarrow t) \in \mathcal{R}$, an occurrence $o \in \mathcal{O}(t)$, and a ground substitution θ such that $s[o] = \theta(s')$ and $t = s[o \leftarrow \theta(t')]$. Note that we define the relation $\rightarrow_{\mathcal{R}}$ only on variable free terms. This is not a serious restriction, since variables could be introduced as function symbols of rank 0. For the rest of the paper we fix a term rewriting system $\mathcal{R}$.

2.2 Hypergraph rewriting systems

The notion of a *hypergraph* is a simple generalization of the notion of an ordinary graph. In contrast to ordinary graphs, a *hyperedge* e of a hypergraph can be connected to an arbitrary finite number of *nodes*. This is formalized by assigning two finite (possibly empty) lists of nodes to e, namely the list of source nodes and target nodes of e, respectively, (for an ordinary graphs, each of these two lists has exactly one member). Moreover, every e is labeled by a symbol λ from a fixed alphabet Γ. Since we associate two list of vertices with e, we assign a pair (n, m) of natural numbers to every $\lambda \in \Gamma$ and require that the list of source (target) nodes of e has length n (m).

Definition 1. A *doubly ranked alphabet* Γ is a finite set of symbols together with a mapping $type_\Gamma : \Gamma \rightarrow \mathbb{N} \times \mathbb{N}$. Let Γ be a doubly ranked alphabet. A *Γ–hypergraph* G, or briefly *hypergraph*, is a tuple $G = (V, E, lab, \sigma, \tau)$, where (i) V and E are finite sets of *nodes* and *hyperedges* (briefly *edges*), respectively, (ii) $lab : E \rightarrow \Gamma$ is a labeling–function and (iii) σ and τ are functions $\sigma, \tau : E \rightarrow V^*$ such that $type_\Gamma(lab(e)) = (|\sigma(e)|, |\tau(e)|)$. The components of a hypergraph G will be denoted by $V_G, E_G, lab_G, \sigma_G$ and τ_G. $\mathcal{H}_\Gamma$ is the class of all Γ–hypergraphs. In most cases we omit the subscript Γ.

We use the following conventions for drawing hypergraphs. Circles represent nodes. We label the circle, which represents the node v, explicitly with v if this is useful for further considerations. A hyperedge e is represented as a box labeled with $lab_G(e)$. All nodes in $\sigma_G(e)$ $(\tau_G(e))$ are pictured below (above) e [1] and are connected with e by a line. The order of the nodes in $\sigma_G(e)$ and $\tau_G(e)$ is represented by their left–to–right order.

Definition 2. Let $G \in \mathcal{H}$. The relation $\rightsquigarrow_G \subseteq V_G \times V_G$ is defined as follows. $v \rightsquigarrow_G w$ iff there exists an $e \in E_G$ such that $v \in \sigma_G(e)$ and $w \in \tau_G(e)$. G is called *acyclic* iff the transitive closure $\rightsquigarrow_G^+$ of $\rightsquigarrow_G$ is irreflexive. For $v \in V_G$ we define the indegree and outdegree of v by $outdeg_G(v) = |\{(e, i) \mid \sigma_G(e)_i = v\}|$ and $indeg_G(v) = |\{(e, i) \mid \tau_G(e)_i = v\}|$, respectively.

[1] Since we will never have to draw cyclic graphs, this can always be achieved.

Notions like the disjoint union of two hypergraphs and subhypergraphs of a hypergraph are defined in the obvious way. Structure preserving maps between hypergraphs are called hypergraph morphisms. They are defined as follows.

Definition 3. Let $G, H \in \mathcal{H}$. A *hypergraph morphism*, or briefly *morphism*, $f : G \to H$ between G and H is a pair (f_V, f_E) of functions $f_V : V_G \to V_H$ and $f_E : E_G \to E_H$ such that for every $e \in E_G$, (i) $lab_H(f_E(e)) = lab_G(e)$, (ii) $\sigma_H(f_E(e)) = f_V^*(\sigma_G(e))$ and (iii) $\tau_H(f_E(e)) = f_V^*(\tau_G(e))$. f is *injective (surjective, bijective)* iff f_V and f_E are injective (surjective, bijective). For $v \in V_G$ and $e \in E_G$, we write $f(v)$ and $f(e)$ instead of $f_V(v)$ and $f_E(e)$, respectively.

As usual, we will identify two hypergraphs if there exists a bijective morphism between them. The hypergraphs that appear in this paper are of the following special form.

Definition 4. A (Γ, m, n)*–net*, or briefly *net*, is a triple $(G, \underline{v}, \underline{w})$, where $G \in \mathcal{H}_\Gamma$ such that $indeg_G(v), outdeg_G(v) \leq 1$ for every $v \in V_G$ and $\underline{v}$ and $\underline{w}$ are repetition free lists of all nodes in $\{v \in V_G \mid indeg_G(v) = 0\}$ and $\{w \in V_G \mid outdeg_G(w) = 0\}$, respectively, where $|\underline{v}| = m$ and $|\underline{w}| = n$. The class of all (Γ, m, n)–nets is denoted by $\mathcal{N}_\Gamma^{(m,n)}$. The net $(G, \underline{v}, \underline{w})$ is briefly denoted by G, whereas the two lists $\underline{v}$ and $\underline{w}$ are denoted by $bottom(G)$ and $top(G)$, respectively. The set of all $G \in \mathcal{N}_\Gamma^{(n,m)}$ such that G is acyclic is denoted by $\mathcal{AN}_\Gamma^{(n,m)}$. If the subscript (m, n) is not important we omit it.

If $G \in \mathcal{N}_\Gamma$, the order of the nodes in $bottom(G)$ and $top(G)$ is represented by their left–to–right order in pictures. The following definition of net rewriting is a special case of the algebraic double pushout approach, see e.g. [Ehr79].

Definition 5. A *net rewrite rule* η is a pair $\eta = (L, R)$ of elements of $\mathcal{AN}_\Gamma^{(m,n)}$. A *direct derivation* $\mathcal{D}$ from $G \in \mathcal{N}_\Gamma^{(k,l)}$ to $H \in \mathcal{N}_\Gamma^{(k,l)}$ (by η) is a tuple (G, η, g, H), where $g : L \to G$ is an injective morphism and H is constructed from G, g and η as follows. In a first step delete from G every edge in $g(E_L)$ and every node in $g(V_L)$ except the nodes $g(v)$ for $v \in bottom(L) \cup top(L)$, which from the local interface. Let us call the resulting net D. The second step constructs H from the disjoint union of D and R by gluing the nodes $g(bottom(L)_i)$, $bottom(R)_i$ for every $1 \leq i \leq m$ and $g(top(L)_j)$, $top(R)_j$ for every $1 \leq j \leq n$. For every node $v \in V_D \subseteq V_G$ we denote the image of v under the canonical embedding of D into H by $tr_\mathcal{D}(v)$. For $v \in V_G \backslash V_D$ we set $tr_\mathcal{D}(v) = undef$. Finally, we define $bottom(H) = tr_\mathcal{D}^*(bottom(G))$ and $top(H) = tr_\mathcal{D}^*(top(G))$. Instead of (G, η, g, H) we write $G \Rightarrow_{\eta,g} H$ of just $G \Rightarrow_\eta H$ if the specific g is not important.

Note that we assume that the morphism g is injective. It is easy to see that the above construction of H is well defined, i.e, that the constructed H is in fact in $\mathcal{N}_\Gamma^{(k,l)}$, $tr_\mathcal{D}(v)$ is defined for every $v \in bottom(G) \cup top(G)$, and that the set of all $v \in V_H$ such that $indeg_H(v) = 0$ is exactly the set $tr_\mathcal{D}^*(bottom(G))$ (analogously for $top(G)$). In particular, due to the restriction on nets, the so called gluing conditions (see the algebraic double pushout approach) are always satisfied. Isolated nodes in the right–hand side R of a rule, i.e., nodes $v \in bottom(R) \cap top(G)$ may result in the gluing of nodes in G. For instance, the effect of the rule in Figure 1 is the deletion of the images of the two edges in the left–hand side and the gluing of (the image of) v_i with (the image of) w_i for every $1 \leq i \leq n$ (similarly for the rule in Figure 2).

Definition 6. A *net rewriting system* is a set $\mathcal{HR}$ of net rewrite rules. Let $G, H \in \mathcal{N}_\Gamma$. A *derivation* $\mathcal{D}$ from G to H by $\mathcal{HR}$ is either a bijective morphism $f : G \to H$ or a pair, consisting of a direct derivation $\mathcal{D}_1 = (G \Rightarrow_\eta G_1)$, where $\eta \in \mathcal{HR}$, and a derivation $\mathcal{D}_2$ from G_1 to H by $\mathcal{HR}$. In the first case we define $tr_\mathcal{D} = f$. In the second case we define $tr_\mathcal{D} = tr_{\mathcal{D}_2} \circ tr_{\mathcal{D}_1}$. We write $G \Rightarrow^*_{\mathcal{HR}} H$ iff there exists a derivation from G to H by $\mathcal{R}$. If $\mathcal{D}_1$ is a derivation from G_1 to G_2 by $\mathcal{HR}$ and $\mathcal{D}_2$ is a derivation from G_2 to G_3 by $\mathcal{HR}$, then the derivation $\mathcal{D}_1\mathcal{D}_2$ from G_1 to G_3 by $\mathcal{HR}$ is defined in the obvious way.

3 Term rewriting as local graph rewriting

This Section contains the main results of this paper. In Section 3.1 we associate to our fixed but arbitrary term rewriting system $\mathcal{R}$ a net rewriting system $\mathcal{H}(\mathcal{R})$. In Section 3.2 (3.3) we prove that $\mathcal{H}(\mathcal{R})$ simulates $\mathcal{R}$ in a sound (complete) way. The use of hypergraphs instead of ordinary graphs will make the technical details of these proofs easier but it has no fundamental reasons.

3.1 Graph rewrite rules for simulating term rewriting

As already noted in the introduction, the sharing and discarding of terms has to be made explicitly in our local approach. For this purpose, we introduce two special edge labels $\bigtriangledown$ and $\top$. Furthermore, for each label λ, we need a kind of inverse label $\overline{\lambda}$.

Definition 7. Let $\overline{\Sigma} = \{\overline{f} \mid f \in \Sigma\}$ be a copy of Σ such that $\Sigma \cap \overline{\Sigma} = \emptyset$. Furthermore, let $\bigtriangledown, \bigtriangleup, \top$, and $\bot$ be new symbols different from all symbols in $\Sigma \cup \overline{\Sigma}$. Define $\Gamma = \Sigma \cup \overline{\Sigma} \cup \{\bigtriangledown, \bigtriangleup, \top, \bot\}$, $\overline{\bigtriangleup} = \bigtriangledown$, $\overline{\top} = \bot$, and $\overline{\overline{\lambda}} = \lambda$ for $\lambda \in \Gamma$. We extend the alphabet Γ to a doubly ranked alphabet $(\Gamma, type)$, where $type : \Gamma \to \mathbb{N} \times \mathbb{N}$ is defined by (i) $type(f) = (n, 1)$ for every $f \in \Sigma^{(n)}$, (ii) $type(\bigtriangledown) = (1, 2)$, (iii) $type(\top) = (1, 0)$ and (iv) $type(\overline{\lambda}) = (n, m)$ iff $type(\lambda) = (m, n)$. In the following, we will denote the doubly ranked alphabet $(\Gamma, type)$ just by Γ. For $G \in \mathcal{N}_\Gamma^{(m,n)}$, the *inverse* $\overline{G} \in \mathcal{N}_\Gamma^{(n,m)}$ of G is defined by $\overline{G} = (V_G, E_G, \overline{lab}, \overline{\sigma}, \overline{\tau})$, where $\overline{lab}(e) = \overline{lab_G(e)}$, $\overline{\sigma}(e) = \tau_G(e)$ and $\overline{\tau}(e) = \sigma_G(e)$ for every $e \in E_G$ and $top(\overline{G}) = bottom(G)$, $bottom(\overline{G}) = top(G)$. For a net rewrite rule $\eta = (L, R)$ we define $\overline{\eta} = (\overline{L}, \overline{R})$. We use the abbreviation $\mathcal{J}^{(m,n)} = \mathcal{AN}^{(m,n)}_{\Sigma \cup \{\bigtriangledown, \top\}}$, where we omit the subscript (m, n) if it is not important.

For every $f \in \Sigma^{(n)}$, we define net rewrite rules by Figures 1 - 8, see also [Laf95] for similar rules in the framework of two dimensional rewriting. The following remarks will help to understand these rules.

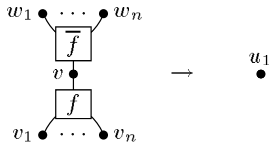


Fig. 1. $\mathcal{H}(f, \overline{f})$

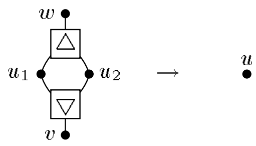


Fig. 2. $\mathcal{H}(\bigtriangledown, \triangle)$

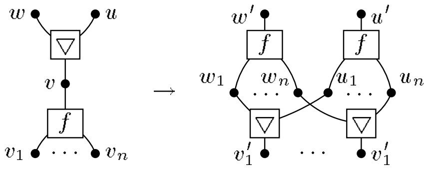


Fig. 3. $\mathcal{H}(f, \bigtriangledown)$

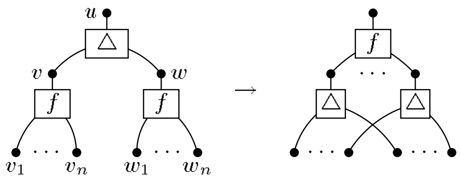


Fig. 4. $\mathcal{H}(f, \triangle)$

The rule $\mathcal{H}(f, \overline{f})$ defines $\overline{f}$–labeled edges as destructors, which select the arguments $s_1, \ldots, s_n$ of a term $f(s_1, \ldots, s_n)$. $\bigtriangledown$–labeled edges act as duplicators (see also[Lam90], [Laf90], [Laf95]). A $\triangle$–labeled edge has a contrary effect as the rules $\mathcal{H}(\bigtriangledown, \triangle)$ and $\mathcal{H}(f, \triangle)$ show. It merges the trees, rooted at its two source nodes, provided that they are equal trees (otherwise it blocks). Thus, the effect of these rules is similar to the folding rules in [HP91]. But in contrast to [HP91], where the folding of identical jungles must be performed in order to warrant the applicability of all graph rewrite rules and thus may lead to unnecessary work, in our approach the folding of identical subtrees is performed on demand, i.e., after the application of a rule that corresponds to a single term rewrite step (see also Example 1). The rules $\mathcal{H}(f, \top)$ and $\mathcal{H}(\bigtriangledown, \top)$ perform garbage collection. These rules are not crucial for soundness and completeness. Finally, the rules $\mathcal{H}(\bigtriangledown, \bigtriangledown)$ and $\mathcal{H}(\bigtriangledown)$ specify a kind of associativity and commutativity for $\bigtriangledown$–labeled edges. Again, these rules are not crucial for soundness and completeness, but together with rule $\mathcal{H}(\bigtriangledown, \triangle)$ they may lead to shorter derivations, see e.g. Example 1.

For every rule η, mentioned above, we could add the rules $\overline{\eta}$ to our system. The soundness result of Theorem 15 would still hold. Furthermore, this would increase the symmetry of our system. In order to keep the system easy to survey, we have decided to omit these rules.

It is easy to see that the set consisting of the rules in Figures 1 - 6 forms a locally confluent and terminating set of rules on $\mathcal{AN}_\Gamma$. The only two critical pairs that arise, namely between $\mathcal{H}(f, \bigtriangledown), \mathcal{H}(\bigtriangledown, \triangle)$ and $\mathcal{H}(f, \bigtriangledown), \mathcal{H}(\bigtriangledown, \top)$, respectively, can easily be resolved. Termination holds, because every rule decreases the sum of the lengths of all different maximal $\leadsto_G$–paths that lead to nodes of the form $\sigma_G(e)$ for $lab_G(e) \in \overline{\Sigma} \cup \{\bigtriangledown\}$ or $\tau_G(e)$ for $lab_G(e) \in \{\triangle, \top\}$ (of course, this holds only for acyclic nets). The rule $\mathcal{H}(\bigtriangledown)$ can implicitly be included by adding the rules that result from $\mathcal{H}(\bigtriangledown, \triangle)$ and $\mathcal{H}(\bigtriangledown, \top)$ by interchanging the two target nodes of the $\bigtriangledown$–labeled edge. The resulting system is

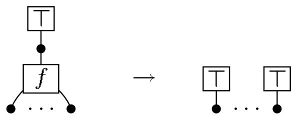


Fig. 5. $\mathcal{H}(f, \top)$

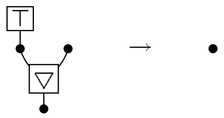


Fig. 6. $\mathcal{H}(\bigtriangledown, \top)$

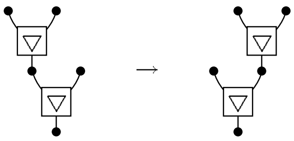


Fig. 7. $\mathcal{H}(\bigtriangledown, \bigtriangledown)$

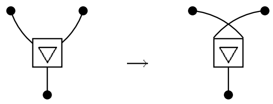


Fig. 8. $\mathcal{H}(\bigtriangledown)$

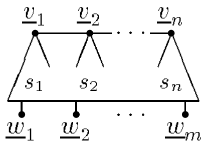
still locally confluent and terminating on $\mathcal{AN}_\Gamma$. Such a simple solution is not possible for the associativity rule $\mathcal{H}(\bigtriangledown, \bigtriangledown)$.

Note that the rules in Figures 1 - 8 only depend on the alphabet Σ but not on the specific term rewrite rules in $\mathcal{R}$. In the next definitions we specify the $\mathcal{R}$–dependent part. This part contains for every term rewrite rule $(s \to t) \in \mathcal{R}$ exactly one net rewrite rule $\mathcal{H}(s \to t)$. This rule has the remarkable feature that it replaces only a single hyperedge. Such rules are called hyperedge replacement rules and are well studied in the literature (see [DHK] for an overview).

Definition 8. Let $G \in \mathcal{AN}_\Gamma^{(l,m)}$, $H \in \mathcal{AN}_\Gamma^{(m,n)}$ The *composition of* G *and* H denoted by $G \circ H \in \mathcal{AN}_\Gamma^{(l,n)}$, results from the disjoint union of G and H by gluing the nodes $top(G)_i$ and $bottom(H)_i$ for every $1 \leq i \leq m$. Let g be the canonical injective embedding of G into $G \circ H$. If $v \in V_G$, we denote the node $g(v) \in G \circ H$ by v as well. An analogous abbreviation is used for nodes $w \in V_H$. Finally, we define $bottom(G \circ H) = bottom(G)$ and $top(G \circ H) = top(H)$.

Definition 9. Let $G \in \mathcal{J}^{(m,n)}$ and $v \in V_G$. The number $height_G(v) \in \mathrm{I\!N}$ and the term $term_G(v) \in T_\Sigma(bottom(G))$ are defined inductively by the following two conditions. (i) If $indeg_G(v) = 0$, i.e., $v \in bottom(G)$ then $height_G(v) = 0$ and $term_G(v) = v$. (ii) Let $indeg_G(v) = 1$ and let $e \in E_G$ be the unique edge such that $v \in \tau_G(e)$. Then $height_G(v) = 1 + max\{height_G(w) \mid w \in \sigma_G(e)\}$ (we define $max(\emptyset) = 0$). Let $\sigma_G(e) = v_1 \dots v_n$. If $lab_G(e) = \bigtriangledown$ (this implies $n = 1$) then $term_G(v) = term_G(v_1)$. If $lab_G(e) = f \in \Sigma$ then $term_G(v) = f(term_G(v_1), \dots, term_G(v_n))$.

Finally, we say that G represents the list $term_G^*(top(G))$ of terms in $T_\Sigma(bottom(G))$ and we picture G as shown on the right, where we use the abbreviation $\underline{w} = bottom(G)$ and $\underline{v} = top(G)$. The inverse $\overline{G}$ of G is pictured as G upside–down. For instance, the right net in Figure 9 shows a net of the form $\overline{G} \circ H$, where $G \in \mathcal{AN}_\Gamma^{(m,n)}$ and $H \in \mathcal{AN}_\Gamma^{(m,1)}$.



Definition 10. For every $(s \to t) \in \mathcal{R}$ with $s = f(s_1, \dots, s_n)$, the hyperedge replacement rule $\mathcal{H}(s \to t) = (L_{s \to t}, R_{s \to t})$ consists of the following components.

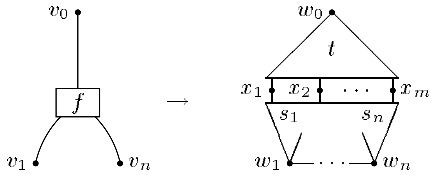
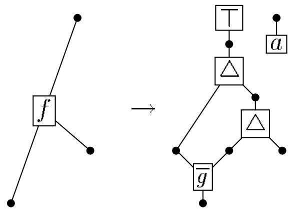


Fig. 9. $\mathcal{H}(s \to t)$ where $Var(s) = \{x_1, \ldots x_m\}$ and $s = f(s_1, \ldots, s_n)$

(i) $L_{s\to t} = ((\{v_0, \ldots, v_n\}, e, lab, \sigma, \tau), v_1 \ldots v_n, v_0)$, where $lab(e) = f$, $\sigma(e) = v_1 \ldots v_n$ and $\tau(e) = v_0$. (ii) $R_{s\to t} = \overline{R_1} \circ R_2$, where $R_1 \in \mathcal{J}^{(m,n)}$ represents $s_1 \ldots s_n$ and $R_2 \in \mathcal{J}^{(m,1)}$ represents t such that $bottom(R_1) = bottom(R_2)$ [2]. The rule $\mathcal{H}(s \to t)$ is shown in Figure 9, where we assume $Var(s) = \{x_1, \ldots, x_m\}$. Finally, the net rewriting system $\mathcal{H}(\mathcal{R})$ consists of the net rewrite rules in Figure 1 - 8 and the rules in $\{\mathcal{H}(s \to t) \mid (s \to t) \in \mathcal{R}\}$.

The nodes in $bottom(R_1)$ are exactly the variables that appear in $s_1, \ldots, s_n$, i.e, the variables in $Var(s)$, whereas $bottom(R_2) = Var(t)$. Since $Var(t)$ may be a proper subset of $Var(s)$, it may be necessary to introduce the variables from $Var(s)\backslash Var(t)$ as dummy bottom nodes into R_2 via $\top$–labeled edges in order to achieve $bottom(R_1) = bottom(R_2)$, see e.g. the following example, where R_2 consists of the two edges labeled with a and $\top$, respectively.

Example 1. Let $r = f(g(x, x), x) \to a$. The picture one the right shows the rule $\mathcal{H}(r)$. Two derivations by $\mathcal{H}(\{r\})$, starting from a $G \in \mathcal{J}^{(0,1)}$ that represents the term $f(g(a, a), a)$ are shown in Figure 10. In the first one, the rules $\mathcal{H}(r)$, $\mathcal{H}(g, \overline{g})$, $\mathcal{H}(a, \bigtriangledown)$, $\mathcal{H}(a, \bigtriangledown)$, $\mathcal{H}(a, \bigtriangleup)$, $\mathcal{H}(a, \bigtriangleup)$, $\mathcal{H}(a, \top)$ are applied in this order. In the second derivation, the rules $\mathcal{H}(r)$, $\mathcal{H}(g, \overline{g})$, $\mathcal{H}(\bigtriangledown, \bigtriangledown)$, $\mathcal{H}(\bigtriangledown, \bigtriangleup)$, $\mathcal{H}(\bigtriangledown, \bigtriangleup)$, $\mathcal{H}(a, \top)$ are applied in this order.



3.2 Soundness of local graph rewriting

In this section, we prove the soundness of the net rewriting system $\mathcal{H}(\mathcal{R})$. The following lemma can be proven by an inspection of the rules in $\mathcal{H}(\mathcal{R})$.

Lemma 11. Let $G \in \mathcal{AN}_{\Gamma}^{(m,n)}$ and let $\mathcal{D}$ be a direct derivation from G to H by $\mathcal{H}(\mathcal{R})$. Then $H \in \mathcal{AN}_{\Gamma}^{(m,n)}$.

[2] Of course, R_1 and R_2 are not uniquely determined. However, the concrete choice does not influence soundness or completeness, but efficiency. One should try to exploit as much sharing as possible in R_1 and R_2.

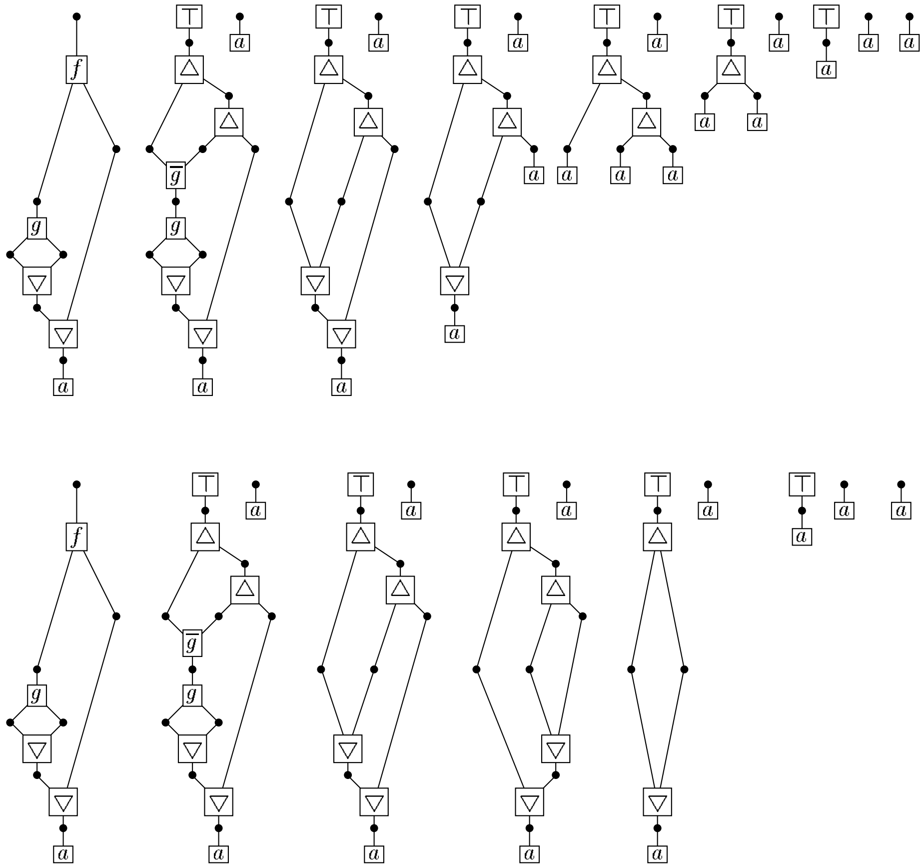


Fig. 10. Two derivations by $\mathcal{H}(\{f(g(x,x),x) \to a\})$

The main idea for the proof of the soundness theorem is to assign ground terms to the nodes of a net G (this is the point where the use of hypergraphs instead of ordinary graphs simplifies some technical details) such that the following local conditions are satisfied.

Definition 12. Let $G \in \mathcal{N}_\Gamma$. A function $M : V_G \to T_\Sigma$ is a *model of* G iff for every $e \in E_G$ and $f \in \Sigma^{(n)}$ the following conditions hold.

- If $lab_G(e) = f$, then $f(M(\sigma_G(e)_1), \ldots, M(\sigma_G(e)_n)) \to^*_{\mathcal{R}} M(\tau_G(e))$.
- If $lab_G(e) = \overline{f}$, then $M(\sigma_G(e)) \to^*_{\mathcal{R}} f(M(\tau_G(e)_1), \ldots, M(\tau_G(e)_n))$.
- If $lab_G(e) \in \{\bigtriangledown, \bigtriangleup\}$, then $M(v) \to^*_{\mathcal{R}} M(w)$ for every $v \in \sigma_G(e)$, $w \in \tau_G(e)$.

Lemma 13. If $G \in \mathcal{J}$, M is a model of G, and M' is a model of the inverse $\overline{G}$ of G then

(1) $\forall v \in V_G : term_G(v)\{M(x)/x \mid x \in bottom(G)\} \rightarrow^*_{\mathcal{R}} M(v)$ and

(2) $\forall v \in V_G : M'(v) \rightarrow^*_{\mathcal{R}} term_G(v)\{M'(x)/x \mid x \in bottom(G)\}$.

Proof. We only prove the first statement. Since $G \in \mathcal{J}$, the statement can be proven by an induction on $height_G(v)$. We will use the abbreviation $\{*\}$ for the substitution $\{M(x)/x \in bottom(G)\}$. If $height_G(v) = 0$, i.e., $v \in bottom(G)$ then $M(v) = v\{M(v)/v\} = term_G(v)\{*\}$. Assume that $height_G(v) > 0$. Thus, there exists a unique edge $e \in E_G$ such that $v \in \tau_G(e)$. We separate to cases.
case 1: $lab_G(e) = f \in \Sigma^{(n)}$, thus $\tau_G(e) = v$: Let $\sigma_G(e) = v_1 \dots v_n$. We obtain

$$\begin{aligned} term_G(v)\{*\} &= f(term_G(v_1), \dots, term_G(v_n))\{*\} \\ &= f(term_G(v_1)\{*\}, \dots, term_G(v_n)\{*\}) \\ &\rightarrow^*_{\mathcal{R}} f(M(v_1), \dots, M(v_n)) && \text{(induction hypothesis)} \\ &\rightarrow^*_{\mathcal{R}} M(v) && (M \text{ is a model of } G). \end{aligned}$$

case 2: $lab_G(e) = \bigtriangledown$: Let $\sigma_G(e) = w$. Then $term_G(v)\{*\} = term_G(w)\{*\} \rightarrow^*_{\mathcal{R}} M(w) \rightarrow^*_{\mathcal{R}} M(v)$. □

Lemma 14. Let $\mathcal{D}$ be a direct derivation from $G \in \mathcal{N}_\Gamma$ to $H \in \mathcal{N}_\Gamma$ by $\eta \in \mathcal{H}(\mathcal{R})$ and let N be a model of H. Then there exists a model M of G such that for every $v \in V_G$, if $tr_{\mathcal{D}}(v) \neq undef$ then $M(v) = N(tr_{\mathcal{D}}(v))$.

Proof. Since $\mathcal{H}(\mathcal{R})$ consists of nine different types of hypergraph rewrite rules, we have to separate nine different cases for the rule $\eta = (L, R)$. First, we have to find a morphism $g : L \rightarrow G$. Since g must be injective, we can denote the image $g(v) \in V_G$ of $v \in V_L$ by v as well. Analogously, the image of a node $w \in V_R$ in H can be denoted by w as well. Therefore, in order to denote nodes of G and H, we can use the additional node labels in Figure 1 - 9, which were only introduced for this purpose.

If $tr_{\mathcal{D}}(v) \neq undef$, we are forced to set $M(v) = N(tr_{\mathcal{D}}(v))$. Thus, we only have to define $M(v)$ for nodes $v \in V_G$ with $tr_{\mathcal{D}}(v) = undef$, using the values $M(w)$ for nodes $w \in V_G$ with $tr_{\mathcal{D}}(w) \neq undef$, in such a way that M becomes a model for G. Obviously, it is sufficient to check the conditions for M to be a model of G only for those edges of G that are deleted in the direct derivation $\mathcal{D}$.
case 1: $G \Rightarrow_{\mathcal{H}(f,\overline{f})} H$ (see Figure 1): Only $tr_{\mathcal{D}}(v)$ is not defined. We define $M(v) = f(M(v_1), \dots, M(v_n))$. Since $M(v_i) = N(tr_{\mathcal{D}}(v_i)) = N(u_i) = N(tr_{\mathcal{D}}(w_i)) = M(w_i)$ for every $1 \leq i \leq n$, we obtain a model of G.
case 2: $G \Rightarrow_{\mathcal{H}(\bigtriangledown,\bigtriangleup)} H$ (see Figure 2): This time, the setting $M(u_1) = M(v) = M(u_2)$ gives a model of G.
case 3: $G \Rightarrow_{\mathcal{H}(f,\bigtriangledown)} H$ (see Figure 3): If we define $M(v) = f(M(v_1), \dots, M(v_n))$ we obtain

$$\begin{aligned} M(v) &= f(M(v_1), \dots, M(v_n)) \\ &= f(N(tr_{\mathcal{D}}(v_1)), \dots, N(tr_{\mathcal{D}}(v_n))) = f(N(v'_1), \dots, N(v'_n)) \\ &\rightarrow^*_{\mathcal{R}} f(N(w_1), \dots, N(w_n)) && (N \text{ is a model for } H) \\ &\rightarrow^*_{\mathcal{R}} N(w') = N(tr_{\mathcal{D}}(w)) = M(w) && (N \text{ is a model for } H) \end{aligned}$$

and analogously $M(v) \to^*_{\mathcal{R}} M(u)$.
case 4: $G \Rightarrow_{\mathcal{H}(f,\triangle)} H$ (see Figure 4): We define $M(v) = f(M(v_1), \ldots, M(v_n))$ and $M(w) = f(M(w_1), \ldots, M(w_n))$. A calculation similar to the one in case 3 shows that $M(v) \to^*_{\mathcal{R}} M(u)$ and $M(w) \to^*_{\mathcal{R}} M(u)$. The rules in Figures 5 - 9 can be dealt similarly. The following case is the main case.
case 5: $G \Rightarrow_{\mathcal{H}(s\to t)} H$ (see Figure 9): Obviously, N is also a model of any subnet of H. In particular, N is a model of the subnet that consists of the lower part of the right net in Figure 9, i.e, the image of R_1 in G. Lemma 13.(2) implies $N(tr_{\mathcal{D}}(v_i)) = N(w_i) \to^*_{\mathcal{R}} s_i\{N(x_j)/x_j \mid 1 \le j \le m\}$ for every $1 \le i \le n$. Thus,

$$\begin{aligned}
f(M(v_1), \ldots, M(v_n)) &= f(N(tr_{\mathcal{D}}(v_1)), \ldots, N(tr_{\mathcal{D}}(v_n))) \\
&\to^*_{\mathcal{R}} f(s_1, \ldots, s_n)\{N(x_j)/x_j \mid 1 \le j \le m\} \\
&\to_{\mathcal{R}} t\{N(x_j)/x_j \mid 1 \le j \le m\} \qquad (\, s \to t \in \mathcal{R}) \\
&\to^*_{\mathcal{R}} N(tr_{\mathcal{D}}(v_0)) = M(v_0) \qquad \text{(Lemma 13.(1))},
\end{aligned}$$

□

The following main result of this section is now a simple consequence of the previous lemma.

Theorem 15. Let $\mathcal{D}$ be a derivation from $G \in \mathcal{J}^{(0,1)}$ to $H \in \mathcal{J}^{(0,1)}$ by $\mathcal{H}(\mathcal{R})$. Then $term_G(v) \to^*_{\mathcal{R}} term_H(tr_{\mathcal{D}}(v))$ for every $v \in V_G$ with $tr_{\mathcal{D}}(v) \neq undef$. In particular, $term_G(top(G)) \to^*_{\mathcal{R}} term_H(top(H))$.

Proof. Since $H \in \mathcal{J}^{(0,1)}$, we can define a model N of H by $N(w) = term_H(w) \in T_\Sigma$ for every $w \in V_H$. Let $v \in V_G$ with $tr_{\mathcal{D}}(v) \neq undef$. Let n be the number of direct derivations in the derivation $\mathcal{D}$. After n applications of Lemma 14 we get a model M of G such that $M(v) = N(tr_{\mathcal{D}}(v)) = term_H(tr_{\mathcal{D}}(v))$. Since $G \in \mathcal{J}^{(0,1)}$, i.e., in particular $bottom(G) = \emptyset$, Lemma 13.(1) gives $term_G(v) \to^*_{\mathcal{R}} M(v) = term_H(tr_{\mathcal{D}}(v))$. The second statement is just a specialization of the first statement. □

Note the modularity of the soundness proof. The inclusion of further net rewrite rules leads only to further cases in the proof of Lemma 14.

3.3 Completeness of local graph rewriting

In this section we prove the completeness of the net rewriting system $\mathcal{H}(\mathcal{R})$. In order to prove completeness, we will only use the rules $\mathcal{H}(f, f')$, $\mathcal{H}(f, \bigtriangledown)$, $\mathcal{H}(f, \triangle)$, $\mathcal{H}(f, \top)$, and the rules $\mathcal{H}(s \to t)$ for $(s \to t) \in \mathcal{R}$. This fact will be used in the next section.

Theorem 16. Let $G \in \mathcal{J}^{(0,1)}$ and $term_G(top(G)) = s \to_{\mathcal{R}} t$. Then there exists a derivation $\mathcal{D}$ from G to $H \in \mathcal{J}^{(0,1)}$ by $\mathcal{H}(\mathcal{R})$ such that $t = term_H(tr_{\mathcal{D}}(top(G))) = term_H(top(H))$.

The details of the proof can be found in the appendix. We only give a brief outline of the proof. Since $s \to_{\mathcal{R}} t$, there exists a rule $r = (f(s_1, \ldots, s_n) \to t') \in \mathcal{R}$, an occurrence $o \in \mathcal{O}(s)$, and a ground substitution ϕ such that $s[o] = \phi(f(s_1, \ldots, s_n)) = f(\phi(s_1), \ldots, \phi(s_n))$ and $t = s[o \leftarrow \phi(t')]$. Since $term_G(top(G)) = s$ there exists a unique hyperedge $e \in E_G$ with $lab_G(e) = f$ that corresponds to the occurrence o, but this hyperedge may be shared. Since we want to simulate a single rewrite step instead of a parallel rewrite step, we have to remove all $\bigtriangledown$–labeled edges that appear above $\tau_G(e)$ using the rules $\mathcal{H}(f, \bigtriangledown)$. In the resulting hypergraph G', the edge e' that corresponds to the occurrence o is not shared. Note that $term_{G'}(\sigma_G(e')_i) = \phi(s_i)$. Now we can apply the rule $\mathcal{H}(r)$ to the edge e'. After this step we have introduced new edges, labeled with symbols from $\overline{\Sigma} \cup \{\bigtriangledown, \top\}$, which represent the terms $s_1, \ldots, s_n$. By applying the rules $\mathcal{H}(f, \overline{f}), \mathcal{H}(f, \bigtriangledown)$, $\mathcal{H}(f, \bigtriangleup)$ and $\mathcal{H}(f, \top)$ we can eliminate this edges. The resulting net is H.

4 Interaction nets

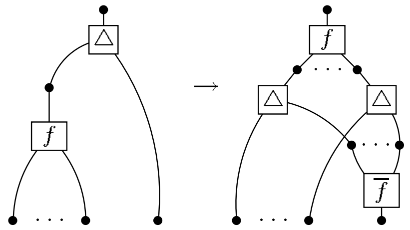
In the following we briefly recall Lafont's interaction nets, see [Laf90] [3]. Our presentation of interaction nets follows closely [Ban95].

Assume a doubly ranked alphabet Γ, where for every $\lambda \in \Gamma$ with $type_\Gamma(\lambda) = (m, n)$, a number $pp(\lambda) \in \{1, \ldots, m+n\}$ is specified. If $G \in \mathcal{N}_\Gamma$ and $e \in E_G$, the nodes in $\sigma_G(e)$ and $\tau_G(e)$ are called the *ports* of e. Moreover, if $lab_G(e) = \lambda$ with $type_\Gamma(\lambda) = (m, n)$ the node $\sigma_G(e)_{pp(\lambda)}$ (iff $1 \le pp(\lambda) \le m$) or $\tau_G(e)_{pp(\lambda)-m}$ (iff $m < pp(\lambda) \le m + n$) is called the *principle port* of e, denoted by $pp_G(e)$. An *interaction net* consists of a set of special net rewrite rules, called *interaction rules*. An *interaction rule* is a net rewrite rule (L, R), where E_L consists of exactly two hyperedges e_1 and e_2, such that $pp_L(e_1) = pp_L(e_2)$. All other ports of e_1 and e_2 must be pairwise distinct and distinct from $pp_L(e_1)$. In an interaction net, for every pair λ_1, λ_2 of different labels, there exists at most one interaction rule whose edges are labeled with λ_1 and λ_2. Together with the restriction of binary interaction, this condition ensures local confluence.

In the last two sections, we have shown that $\mathcal{R}$ can be simulated by the set of hyperedge replacement rules $\{\mathcal{H}(s \to t) \mid (s \to t) \in \mathcal{R}\}$ plus the rules $\mathcal{H}(f, \bigtriangleup)$, $\mathcal{H}(f, \overline{f})$, $\mathcal{H}(f, \bigtriangledown)$, and $\mathcal{H}(f, \top)$ for every $f \in \Sigma$. If we set $pp(f) = n + 1$ for $f \in \Sigma^{(n)}$ and $pp(\overline{f}) = pp(\bigtriangledown) = pp(\top) = 1$, the last three rules are obviously interaction rules. If we set $pp(\bigtriangleup) = 1$, the rule $\mathcal{H}(f, \bigtriangleup)$ can be simulated by the interaction rule shown on the next page. Thus, we can simulate an arbitrary term rewriting system by a set of hyperedge replacement rules plus an interaction net. Therefore, the combination of these two formalisms seems to be a very powerful model of computation. Interaction nets can simulate deterministic Turing machines ([Laf92]), but due to the confluence property, only deterministic computations can be performed.

[3] In fact, due to the restriction on the set $\mathcal{N}_\Gamma$, our definition of interaction nets is a typed one in Lafont's sense, where we have exactly one type and the source (target) nodes of an edge e are the input (output) ports of e.

On the other hand, hyperedge replacement systems are inherently non–deterministic but far away from being Turing–complete. In a certain sense, the enrichment of interaction nets with hyperedge replacement rules seems to be the mildest possible extension in order to introduce non–determinism into the formalism of interaction nets. In a derivation of such a combined system,



applications of hyperedge replacement rules are the only points where a non–deterministic choice may appear, whereas a sequence of binary interactions corresponds to a (possibly non–terminating) deterministic calculation. From another point of view, hyperedge replacement rules may be seen as (possibly recursive) procedure definitions which substitute an agent (the left–hand side of the rule) by its definition (the right–hand side of the rule), which consists of a network of agents.

5 Conclusion

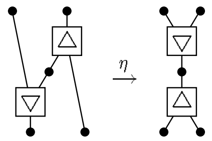
In this paper we have simulated an arbitrary term rewriting system $\mathcal{R}$ by a net rewriting system $\mathcal{H}(\mathcal{R})$ that splits into a $\mathcal{R}$–dependent set of hyperedge replacement rules, one for each rule in $\mathcal{R}$, and a $\mathcal{R}$–independent interaction net. The following points are worthy to note.

Our net–representation of term rewriting can be seen as a refinement of the approach in [CR93] in the sense that a single hyperedge replacement jungle rewrite step corresponds to a sequence of local net rewrite steps in our approach. Such a locally operating refinement seems to be suitable for a parallel implementation. Moreover, since the sizes of the left–hand sides of rules in $\mathcal{H}(\mathcal{R})$ is independent of $\mathcal{R}$, every rewrite step needs approximately the same amount of work for checking applicability. Thus, the number of of rewrite steps, performed in $\mathcal{H}(\mathcal{R})$ is more significant for cost estimates than the number of rewrite steps in $\mathcal{R}$.

Due to the explicit sharing by $\bigtriangledown$–labeled edges we are able to simulate an arbitrary term rewriting system in a sound and complete way. This does not hold for the approaches presented in [CR93] and [HP91], where, due to the implicit sharing of common subterms, term rewriting systems like $\{f(x) \to g(x,x), a \to b, g(a,b) \to c\}$ cannot be simulated in the strong sense required in Theorem 16. Only a weaker notion of completeness, namely completeness w.r.t applicability can be proven in the works mentioned above. Moreover, our approach incorporates garbage collection via $\top$–labeled edges.

As already mentioned in the introduction, the encoding of term rewriting systems into interaction nets was also studied in [FM96a]. In this paper it is shown that the class of orthogonal, constructor–based and match–sequential term rewriting systems can be encoded into pure interaction nets. Furthermore, an extension of the interaction net framework is proposed in which every orthogonal constructor–based term rewriting systems can be simulated. This extension is based on the notion of a state. Via the state, several parallel working interaction nets can communicate. In contrast to our extension by hyperedge replacement rules, the concept of binary interaction is maintained in [FM96a]. On the other hand, our extension is able to simulate every term rewriting system.

A point of criticism at our approach could be the possibility of deadlocks. More precisely, it may be possible to apply the hyperedge replacement rule $\mathcal{H}(s' \to t)$ to a jungle–net G that represents the term s, although $s' \to t$ cannot be applied to s. The resulting net G' can be transformed via some interaction rules to a net H, where for example the target node of a f–labeled edge and the source node of a $\overline{g}$–labeled edge ($f \neq g$) are identical. This configuration will stay forever. Moreover, the work that led from G to H is irreversible. This problem can be fixed by the introduction of further agents and a more complex translation of term rewriting rules into hyperedge replacement rules. In order to keep the presentation in this paper clear and easy, we omitted this extension and will present it in a future work.



Another problem may arise if there exist two edges e_1 and e_2 such that for instance $lab_G(e_1) = \bigtriangledown$, $lab_G(e_2) = \triangle$, $t_G(e_1)_2 = s_G(e_2)_1$ but $t_G(e_1)_1 \neq s_G(e_2)_2$. Such a configuration forces the duplication of the net rooted at $s_G(e_1)$ in order to allow e_2 to move down. The right rule η, whose soundness can be proven in the usual way can resolve this configuration directly and therefore makes the duplication unnecessary. In fact, if only the weaker notion of completeness w.r.t. applicability is required (or attention is restricted to the class of graph reducible term rewriting systems) the inclusion of the rule η together with the rule $\overline{\mathcal{H}(f, \triangle)}$ makes the the duplication rule $\mathcal{H}(f, \bigtriangledown)$ superfluous. On the other hand, the inclusion of η and $\overline{\mathcal{H}(f, \triangle)}$ in favour of $\mathcal{H}(f, \bigtriangledown)$ has several drawbacks. The associativity rule $\mathcal{H}(\bigtriangledown, \bigtriangledown)$ becomes crucial for completeness. Thus, termination of the $\mathcal{R}$–independent part does not hold any longer. Moreover, an application of η may lead to cyclic nets, namely if there exists a path from $t_G(e_1)_1$ to $s_G(e_2)_2$ in the configuration above. Finally, we would leave the interaction net framework. For these reasons, we have decided to omit η from our system.

Some further possible applications for our local approach could be *logic programming* (see [CRPP91], [CR93] for a simulation of logic programs by jungle rewriting), *term narrowing* ([HP96] and [CW94]), combination with interaction net–based λ–evaluators ([FM96b], [Mac95]), *higher–order rewriting systems* ([AL96]), *cyclic term rewriting*, (see [Ken95]).

References

[AL96] A. Asperti and C. Laneve. Interaction systems II: The practice of optimal reductions. *Theoretical Computer Science*, 159, 1996.

[Ban95] R. Banach. The algebraic theory of interaction nets. Report UMCS-95-7-2, University of Manchester, Department of Computer Science, 1995.

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

[CRPP91] A. Corradini, F. Rossi, and F. Parisi-Presicce. Logic programming as hypergraph rewriting. In S. Abramsky and T. S. E. Maibaum, editors, *CAAP'91*, volume 493 of *Lect. Notes Comput. Sci.* Springer–Verlag, 1991.

[CW94] A. Corradini and D. Wolz. Jungle rewriting: an abstract description of a lazy narrowing machine. In H.J. Schneider and H. Ehrig, editors, *Proc. Graph Transformations in Computer Science*, volume 776 of *Lect. Notes Comput. Sci.*, pages 119–137. Springer–Verlag, 1994.

[DHK] F. Drewes, A. Habel, and H.-J. Kreowski. Hyperedge replacement graph grammars. In G. Rozenberg, editor, *Handbook of Graph Transformations, Vol. I: Foundations, Chapter 2.* World Scientific, to appear.

[Ehr79] H. Ehrig. Introduction to the algebraic theory of graph grammars. In *Proceedings of the 1st International Workshop on Graph-Grammars and Their Application to Computer Science and Biology, Lect. Notes Comput. Sci. 73*, pages 1–69. Springer–Verlag, 1979.

[FM96a] M. Fernández and I. Mackie. From term rewriting to generalised interaction nets. volume 1140 of *Lect. Notes Comput. Sci.*, pages 319–333. Springer–Verlag, 1996.

[FM96b] M. Fernández and I. Mackie. Integrating paradigms via interaction nets. In Manuel M. T. Chakravarty, Yike Guo, and Tetsuo Ida, editors, *Multi-Paradigm Logic Programming, MPLP'96. Proceedings of the JICSLP'96 Post Conference Workshop*, pages 137–146. Technical University of Berlin Report Number 96–28, August 1996.

[HP91] B. Hoffmann and D. Plump. Implementing term rewriting by jungle evaluation. *RAIRO*, 25:445–472, 1991.

[HP96] A. Habel and D. Plump. Term graph narrowing. *Math. Struct. in Comp. Science*, 6:649–676, 1996.

[Ken95] R. Kennaway. Infinitary rewriting and cyclic graphs. *Electronic Notes in Theoretical Computer Science*, 2, 1995.

[Laf90] Y. Lafont. Interaction nets. In *Proceedings of the Seventeenth ACM Symposium on Principles of Programming Languages*, pages 95–108. ACM, ACM Press, January 1990.

[Laf92] Y. Lafont. The paradigm of interaction. In *Proceedings of the Workshop on Concurrency*, pages 276–293. Programming Methodology Group, Göteborg, April 1992.

[Laf95] Y. Lafont. Equational reasoning with 2-dimensional diagrams. In J.P. Jouannaud and H. Comon, editors, *Term Rewriting*, number 909 in Lect. Notes Comput. Sci., pages 170–195. Springer Verlag, 1995.

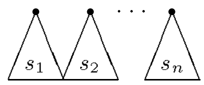
[Lam90] J. Lamping. An algorithm for optimal lambda-calculus reductions. In *Proceedings of the Seventeenth ACM Symposium on Principles of Programming Languages*, pages 16–30. ACM, ACM Press, January 1990.

[Mac95] I. Mackie. A λ-evaluator based on interaction nets. In C. Hankin, I. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods 1994: Proceedings of the Second Imperial College, Department of Computing, Workshop on Theory and Formal Methods*, pages 41–60. Imperial College Press, September 1995.

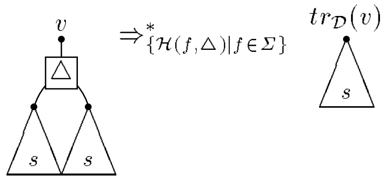
[SPvE93] M.R. Sleep, M.J. Plasmeijer, and M.C.J.D. van Eekelen. *Term graph rewriting - theory and practice.* John Wiley and Sons, 1993.

A Proof of Theorem 16

We define $\mathcal{T}^{(n)} = \{G \in \mathcal{J}^{(0,n)} \mid \forall e \in E_G : lab_G(e) \in \Sigma\}$. Elements of $\mathcal{T}^{(1)}$ are called *trees*. If $G \in \mathcal{T}^{(n)}$ represents $s_1 \dots s_n$, we picture G as shown on the right. Let $G \in \mathcal{T}^{(1)}$ and let e be the unique edge with $\tau_G(e) = top(G)$. For $o \in \mathbb{N}^*$, the edge $G[o] \in E_G$ is inductively defined by (i) $G[\epsilon] = e$ and (ii) $G[io'] = G'[o']$, where $1 \leq i \leq |\sigma_G(e)|$ (otherwise $G[io']$ is undefined) and G' is the subtree rooted at the node $\sigma_G(e)_i$.

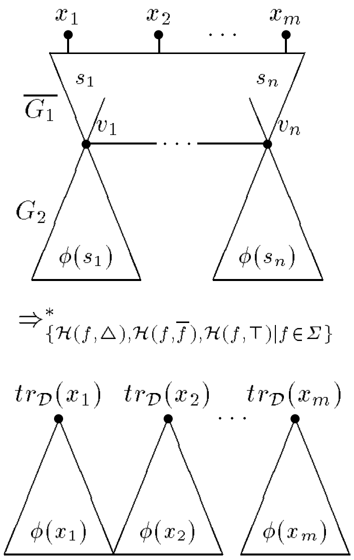


Lemma 17. *Let $s \in T_\Sigma$. Then there exists a derivation by $\{\mathcal{H}(f, \triangle) \mid f \in \Sigma\}$ of the form shown on the right.*



Proof. The lemma can be proven by an induction on the structure of the term s.

Lemma 18.
Let $s_1, \dots, s_n \in T_\Sigma(\{x_1, \dots, x_m\})$ and let ϕ be a ground substitution with $dom(\phi) = \{x_1, \dots, x_m\}$. Then there exists a derivation by $\{\mathcal{H}(f, \triangle), \mathcal{H}(f, \overline{f}), \mathcal{H}(f, \top) \mid f \in \Sigma\}$ of the form shown on the right.



Proof. Denote by G and H the upper and lower net, respectively, in the Figure on the right. G is of the form $G = G_2 \circ \overline{G_1}$, where (i) $G_1 \in \mathcal{J}^{(m,n)}$ represents $s_1 \dots s_n$ and (ii) $G_2 \in \mathcal{T}^{(n)}$ represents $\phi(s_1) \dots \phi(s_n)$. We prove the lemma by an induction on $|E_{G_1}|$. If $|E_{G_1}| = 0$, just note that $n = m$ and that there exists a permutation σ on $\{1, \dots, n\}$ such that $v_{\sigma(i)} = x_i = s_{\sigma(i)}$. Thus, the desired derivation is the identity morphism on G. If $|E_{G_1}| > 0$, we have to separate three different cases.

case 1: There exists an edge $e \in E_{G_1}$ such that $lab_{G_1}(e) = \triangledown$, $\tau_{G_1}(e) = v_i v_j$ with $i \neq j$, and $\sigma_{G_1}(e) = v$. W.l.o.g. we asume $i = 1$ and $j = 2$, see the upper net in Figure 11. Thus, $\phi(s_1) = \phi(term_{G_1}(v_1)) = \phi(term_{G_1}(v_2)) = \phi(s_2)$, i.e, $term_{G_2}(v_1) = term_{G_2}(v_2)$. Therefore, we can apply Lemma 17 to the subnet rooted at node $v \in V_G$ in order to obtain a derivation from G to $H_2 \circ \overline{H_1}$ by $\mathcal{H}(f, \triangle)$, where H_1 results from G_1 by removing the edge e and the nodes v_1 and v_2 and H_2 results from G_2 by merging the two trees, rooted in v_1 and v_2 (both trees represent the same term) to one tree, rooted in v, see the lower net in Figure 11.

Since H_1 has exactly one edge less than G_1, namely e, we can apply the induction hypothesis and get a derivation from $H_2 \circ \overline{H_1}$ to H.
case 2: There exists an edge $e_1 \in E_{G_1}$ such that $lab_{G_1}(e_1) = f \in \Sigma^{(k)}$ and w.l.o.g. $\tau_{G_1}(e_1) = v_1$, see the upper net in Figure 12. Thus, there exist terms $s_{11}, \ldots, s_{1k}$ such that $s_1 = f(s_{11}, \ldots, s_{1k})$. Therefore, $term_{G_2}(v_1) = \phi(s_1) = f(\phi(s_{11}), \ldots, \phi(s_{1k}))$ and there exists a unique edge $e_2 \in E_{G_2}$ with $\tau_{G_2}(e_2) = v_1$ and $lab_{G_2}(e_2) = f$. We can apply the rule $\mathcal{H}(f, \overline{f})$ in order to get a direct derivation from G to $H_2 \circ \overline{H_1}$, where H_1 (H_2) results from G_1 (G_2) by removing the edge e_1 (e_2) and the node v_1, see the lower net in Figure 12. An application of the induction hypothesis yields a derivation from $H_2 \circ \overline{H_1}$ to H.
case 3: There exists an edge $e \in E_{G_1}$ such that $lab_{G_1}(e_1) = \top$ and w.l.o.g. $\tau_{G_1}(e) = v_1$. In this case we can delete the tree rooted at v_1 in G_2 using the rules $\mathcal{H}(f, \top)$. □

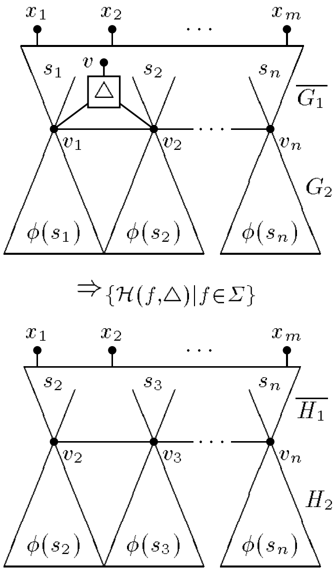


Fig. 11.

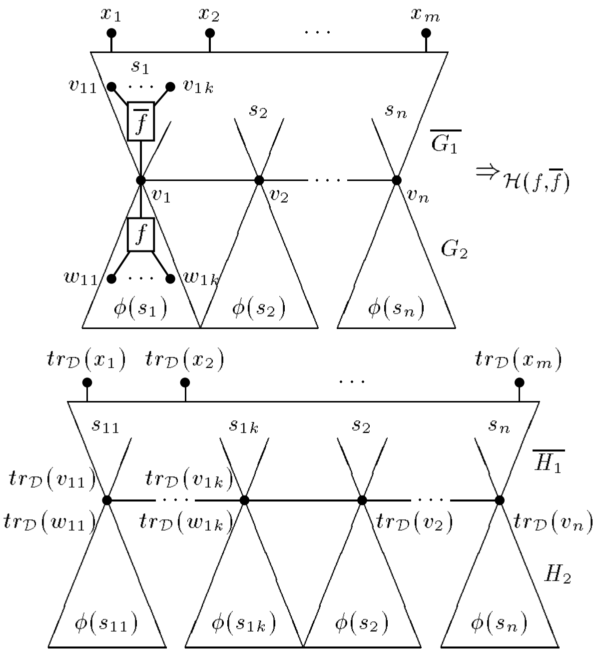
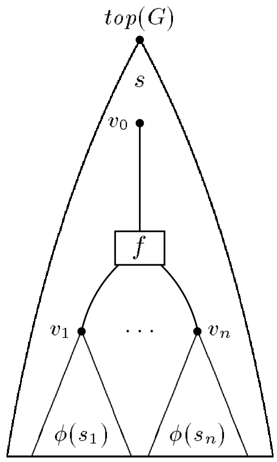


Fig. 12.

Proof for Theorem 16. Since $s \rightarrow_{\mathcal{R}} t$, there exists a rule $r = (f(s_1, \ldots, s_n) \rightarrow t') \in \mathcal{R}$, an occurrence $o \in \mathcal{O}(s)$, and a ground substitution ϕ such that $s[o] = \phi(f(s_1, \ldots, s_n)) = f(\phi(s_1), \ldots, \phi(s_n))$ and $t = s[o \leftarrow \phi(t')]$. Let $e \in E_G$ be the edge that corresponds to the occurrence o and let $\sigma_G(e) = v_1 \ldots v_n$. Thus, $lab_G(e) = f$, and $term_G(v_i) = \phi(s_i)$ for $1 \leq i \leq n$. As noted in the main part of this paper, we can asume that e is not shared. Similarly, we can asume that that no $\bigtriangledown$–labeled edge appears below any v_i for $1 \leq i \leq n$ (the strategy that we follow in this proof is rather inefficient. It would be more efficient to apply the duplication rule $\mathcal{H}(f, \bigtriangledown)$ only if no other rule of the $\mathcal{R}$–independent rules applies and not in advance). Thus, G is of the form shown on the right. To the edge e we can apply the rule $\mathcal{H}(r)$, which gives a direct derivation $\mathcal{D}' = (G \Rightarrow_{\mathcal{H}(r)} G')$. G' is shown in Figure 13. Since G' contains a subnet of the form shown in Figure 9, we can apply Lemma 18 to G', which yields a derivation $\mathcal{D}''$ from G' to G'' with $term_{G''}(tr_{\mathcal{D}''}(x_i)) = \phi(x_i)$ for $1 \leq i \leq m$, see Figure 14. Thus, $term_{G''}(tr_{\mathcal{D}'\mathcal{D}''}(v_0)) = \phi(t')$. Finally, if we set $\mathcal{D} = \mathcal{D}'\mathcal{D}''$ and $H = G''$, we get $term_H(tr_{\mathcal{D}}(top(G))) = s[o \leftarrow \phi(t')] = t$. $\square$



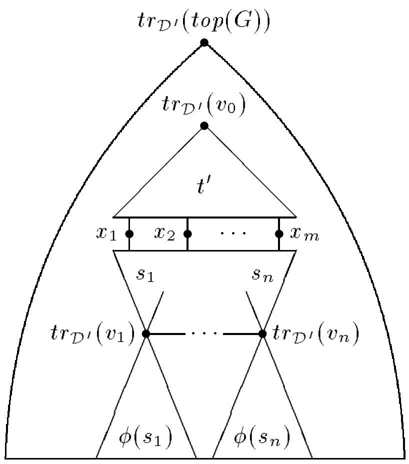


Fig. 13.

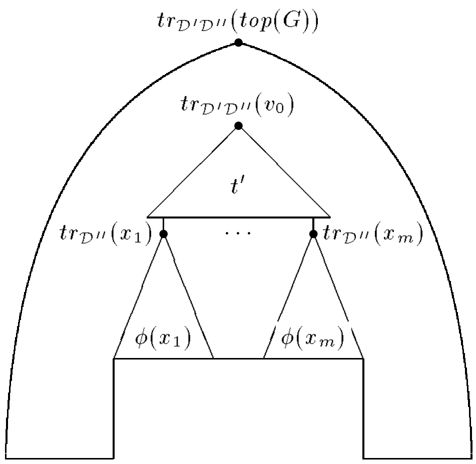


Fig. 14.