Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 72 (2007) 31–44
www.elsevier.com/locate/entcs

Inequational Deduction as Term Graph Rewriting[1]

Andrea Corradini[2] and Fabio Gadducci[2]

Dipartimento di Informatica, Università di Pisa, Italy

Wolfram Kahl[3]

Department of Computing and Software, McMaster University, Canada

Barbara König[4]

Institut für Informatik, Technische Universität München, Germany

Abstract

Multi-algebras allow to model nondeterminism in an algebraic framework by interpreting operators as functions from individual arguments to sets of possible results.
We propose a simple inequational deduction system, based on term graphs, for inferring inclusions of derived relations in a multi-algebra, and we show that term graph rewriting provides a sound and complete implementation of it.

Keywords: Multi-algebra, nondeterminism, inequational deduction system, term-graph rewriting

1 Introduction

The extension of algebraic specification techniques for the treatment of nondeterminism is the topic of numerous contributions to the literature in the last two decades (see [10] for an overview). It is recognised that nondeterminism shows up not only in the specification of intrinsically nondeterministic systems, like concurrent systems, but also when the system is deterministic and one wants to abstract

[1] This work has been partially supported by the Italian MIUR project COMETA (*Computational Metamodels*); and by EU within the FET – Global Computing initiative, project AGILE IST-2001-32747 (*Architectures for mobility*). The funding bodies are not responsible for any use that might be made of the results presented here.

[2] Email: {andrea, gadducci}@di.unipi.it

[3] Email: kahl@cas.mcmaster.ca

[4] Email: koenigb@in.tum.de

doi:10.1016/j.entcs.2002.09.004

away from implementation details or, in general, from information that would be necessary for a complete description of the system (like timing considerations, internals of the states, etc.).

Many extensions of specification techniques to nondeterminism strive for being *non-intrusive* or *conservative*, namely, they try not to change the existing framework for deterministic specifications, and in particular they reduce to the standard theory when only deterministic operations are considered [11]. This fact possibly explains the success of *multialgebras* among the other algebraic approaches, such as e.g. power algebras. Generalising standard partial algebras, in multi-algebras an operator of the signature is interpreted as a function from individual arguments to sets of possible results.

However, the matter concerning deduction systems for (in)equational specifications is not yet fully settled. Problems are of a syntactical nature, and they are tightly related to the so-called *lack of substitutivity*. Roughly, this means that term equivalence is not preserved under the substitution of free variables. This fact suggested to look for alternative presentations of derived operators in a multialgebra, as well as for developing *ad hoc* inference systems.

In [2] a functorial semantics for multialgebras is proposed, extending the classical results concerning algebraic theories. A side result was precisely the characterisation of a simple syntactical device for representing derived operators in a multialgebraic specification, namely, by means of term graphs. In fact the notion of sharing, which is implicit in the framework, allows for an immediate representation both of nondeterminism and domain-restriction, thus obtaining an effective tool for specification in this setting.

In this paper we move a step further, proposing the use of inequations based on term graphs for specifying the inclusion between derived relations in a multi-algebra. We present a deduction calculus for such inequations, proving its soundness (while its completeness is still an open problem). More interestingly, we prove that *term graph rewriting* [9] provides an adequate implementation of inequational deduction, in the sense that $F \sqsubseteq G$ can be deduced if and only if F rewrites to G. This provides an original application of term graph rewriting, which is defined in this paper according to the well-known double-pushout approach in the category of DAG's, even if the format of our rules is more general than the format usually considered in the literature [8].

2 Multi-algebras and relations

The interpretation of a signature operators as relations allows for modeling in an algebraic framework some sort of nondeterministic behaviour [10]. For *multi-algebras*, the relation associated with an operator is a function mapping each input value to a *set* of (possible) output values. Quite obviously, this is equivalent to the usual definition of relations as subsets of the direct product.

Definition 2.1 (relations) *Let A and B be two sets. A* relation $R : A \leftrightarrow B$ *is a function $R : A \to \mathcal{P}(B)$, where $\mathcal{P}$ is the power-set operator. A relation is*

total *if* $|R(a)| \geq 1$ *for all* $a \in A$*; it is* univalent (functional) *if* $|R(a)| \leq 1$ *for all* $a \in A$*. Given two relations* $P, R : A \leftrightarrow B$*,* P *is* included *in* R *(written* $P \subseteq R$*) if* $P(a) \subseteq R(a)$ *for all* $a \in A$*.*

Therefore a univalent relation is a partial function $R : A \rightharpoonup B$, while a univalent and total relation is just a function $R : A \to B$. We shall consider the following operations on relations.

Definition 2.2 (union and composition on relations) *For a set* A*, let* A^i *denote the direct product of* i *copies of* A*. Given two relations* $P : A^j \leftrightarrow A^i$ *and* $R : A^l \leftrightarrow A^k$*, their* union *is the relation* $P \oplus R : A^{j+l} \leftrightarrow A^{i+k}$ *defined as* $P \oplus R(\langle a_1, \ldots a_{j+l}\rangle) = \{\langle b_1, \ldots b_{i+k}\rangle \mid \langle b_1, \ldots b_i\rangle \in P(\langle a_1, \ldots a_j\rangle) \wedge \langle b_{i+1}, \ldots b_{i+k}\rangle \in R(\langle a_{j+1}, \ldots a_{j+l}\rangle)\}$.

Given two relations $P : A \leftrightarrow B$ *and* $R : B \leftrightarrow C$*, their* composition $P; R : A \leftrightarrow C$ *denotes the relation obtained by composing* P *with* $R^+ : \mathcal{P}(B) \to \mathcal{P}(C)$*, the additive extension of* R*.*

For the sake of simplicity, throughout the paper we restrict our analysis to one-sorted signatures, and we denote by Σ an arbitrarily fixed one.

Definition 2.3 (multi-algebras) *A* multi-algebra *A over a (finite) signature* $\Sigma = \biguplus_{i \in \mathbb{N}} \Sigma_i$ *is a pair* $\langle |A|, \rho_A\rangle$*, where the* carrier $|A|$ *is a set, and* $\rho_A = \{f_A \mid f \in \Sigma\}$ *is a family of relations such that, for every* $f \in \Sigma_i$*,* $f_A : |A|^i \leftrightarrow |A|$*.*

Moreover, we define $arity(f) = i$ iff $f \in \Sigma_i$, for all *operators* $f \in \Sigma$.

3 Term Graph Rewriting

This section introduces term graphs and some operations on them following [1], as well as term graph rewriting. To avoid possible confusion, let us anticipate that we shall consider three different kinds of "term graphs":

(i) *Directed acyclic graphs (*DAG*'s)* are the basic ones: we shall introduce the category **DAG** where they live, and "term graph rewriting" will be defined according to the double-pushout approach in this category.

(ii) *Ranked* DAG*'s* are DAG's equipped with chosen root and variable nodes.

(iii) *Term graphs* are isomorphism classes of ranked DAG's.

While DAG's are the objects to be rewritten, their ranked counterparts will be used to define suitable operations over DAG's. In fact, they will be needed to characterise the derived operators of a multialgebraic specification.

Definition 3.1 (directed acyclic graphs and morphisms) *A* directed acyclic graph (DAG) *(over* Σ*) is a triple* $d = \langle N(d), l_d, s_d\rangle$*, where* $N(d)$ *is a set of* nodes*,* $l_d : N(d) \rightharpoonup \Sigma$ *is a partial function called the* labeling function*, and* $s_d : N(d) \rightharpoonup N(d)^*$ *is a partial function called the* successor function*, such that the following conditions are satisfied:*

- $dom(l_d) = dom(s_d)$*;*

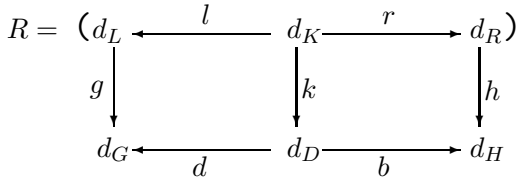


Figure 1. Rewriting step as double-pushout construction.

- *for each node* $n \in dom(l_d)$, $arity(l_d(n)) = length(s_d(n))$;
- d *has no cycles (defined in the expected way).*

A node $n \in N(d)$ *is called* empty *if* $n \notin dom(l_d)$. *We let* $N_\emptyset(d)$ *denote the subset of empty nodes of* $N(d)$, *and* $N_\Sigma(d) = N(d) \backslash N_\emptyset(d)$.

A (DAG) morphism $f : d \to d'$ *is a function* $f : N(d) \to N(d')$ *that preserves labeling and successors; it is an* isomorphism *if in addition* f *is a bijection and also its inverse function* $f^{-1} : N(d') \to N(d)$ *is a* DAG *morphism. We shall denote by* **DAG** *the category having* DAG*'s as objects and* DAG *morphisms as arrows.*

A DAG *morphism* $f : d \to d'$ *is* Σ-injective *if its restriction to* $N_\Sigma(d)$ *is injective; it is* strongly acyclic *if for all* $a_1 \in N_\emptyset(d)$ *and* $a_2 \in N_\Sigma(d)$ *there is no path from* $f(a_1)$ *to* $f(a_2)$ *in* d'; *and it is* plain *if it is both* Σ*-injective and strongly acyclic. It is easy to verify that* Σ*-injective morphisms compose, as well as strongly acyclic ones.*

The rewriting formalism that we shall use is the standard algebraic double-pushout approach [6] in the category of DAG's. We first introduce the general definition of rewriting. Next in Section 5 we shall concentrate on a specific format of rules, for which we shall investigate the adequacy with respect to its representation of inequational deduction.

Definition 3.2 (DPO rewriting in the category of DAG's) *A* (term graph rewriting) rule $R = (d_L \stackrel{l}{\leftarrow} d_K \stackrel{r}{\rightarrow} d_R)$ *is a span of* DAG *morphisms. The* DAG*'s* d_L, d_K, *and* d_R *are called the* left-hand side, *the* interface, *and the* right-hand side *of* R, *respectively. A* (term graph) rewriting system $\mathcal{R}$ *is a set of rules.*

Given a DAG d_G, *a rule* $R = (d_L \stackrel{l}{\leftarrow} d_K \stackrel{r}{\rightarrow} d_R)$, *and a* match *(i.e., a* DAG *morphism)* $g : d_L \to d_G$, *we say that* d_G rewrites to d_H (using R and g) *if the diagram in Figure 1 can be constructed, where both squares are pushouts in category* **DAG**. *In this case,* d_D *is called the* context (DAG), *and we write* $d_G \Rightarrow_{R,g} d_H$, *possibly omitting the subscripts. We write* $d_G \Rightarrow^*_{\mathcal{R}} d_H$ *if there is a possibly empty sequence of rewriting steps from* G *to* H, *using rules in* $\mathcal{R}$.

As usual with DPO presentations, also an operational description of the rewriting mechanism is available. However, it differs from the standard construction holding e.g. in the category of graphs, and for more details we refer the reader to [4], as well as to Lemma 5.4, to be presented later on.

Definition 3.3 (ranked DAG's and term graphs) *A* (j, i)-ranked DAG *is a triple* $g = \langle r, d, v \rangle$, *where* d *is a* DAG *with exactly* j *empty nodes,* $r : \underline{i} \to N(d)$

is a function[5] *called the* root mapping, *and* $v : \underline{j} \to N_\emptyset(d)$ *is a bijection (between* $\underline{j}$ *and the empty nodes of* d*) called the* variable mapping. *If* $g = \langle r, d, v\rangle$ *and* $g' = \langle r', d', v'\rangle$ *are two equally-ranked* DAG*'s, a* ranked DAG morphism $\varphi : g \to g'$ *is a* DAG *morphism* $\varphi : d \to d'$ *such that* $\varphi \circ r = r'$ *and* $\varphi \circ v = v'$.

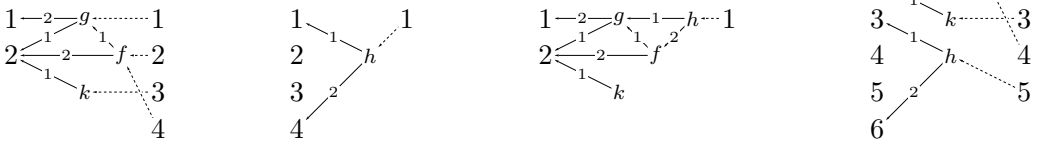
A (j, i)-ranked term graph G *is an isomorphism class of* (j, i)*-ranked* DAG*'s.*

There are two natural operations on term graphs, defined as follows.

Definition 3.4 (composition and union of term graphs) *Let* G^j_k, H^i_j *be term graphs. Their* composition *is the term graph* $G^j_k; H^i_j$ *of rank* (k, i) *obtained by first taking the disjoint union of the graphs underlying* G *and* H*, and then gluing the roots of* G *with the corresponding variables of* H*.*

Let G^i_j, H^k_l *be term graphs. Their* union *is the term graph* $G^i_j \oplus H^k_l$ *of rank* $(j + l, i + k)$ *obtained by first taking the disjoint union of the graphs underlying* G *and* H*, and then concatenating the roots (variables) of* G *with the roots (variables) of* H*.*[6]

The drawing below shows four term graphs. Empty nodes are represented by the natural numbers corresponding to their position in the list of variables, and are depicted as a vertical sequence on the left; nonempty nodes are represented by their labels, from where the edges pointing to the successors leave; the list of numbers on the right represent pointers to the roots: a dashed arrow from j to a node indicates that it is the j-th root. For example, the first term graph G_1 has e.g. rank $(2, 4)$, five nodes (two empty, 1 and 2, and three nonempty, f, g and k), the successors of g are the variables 2 and 1 (in this order), the successors of f are g and 2, the successor of k is 2, and the four roots are g, f, k, and f.



G_1 G_2 $G_3 = G_1 ; G_2$ $G_4 = G_1 \oplus G_2$

These conventions facilitate the understanding of the two basic operations on term graphs. The composition of two term graphs is obtained by first matching the roots of the first term graph with the variables of the second, and then eliminating these variables. Term graph G_3 is e.g. the composition of G_1 and G_2. The union is essentially their disjoint union, where the lists of variables and roots are concatenated. Term graph G_4 is the union of G_1 and G_2.

[5] For a natural number k we assume $\underline{k} = \{1, \ldots, k\}$, thus $\underline{0} = \emptyset$.

[6] Let $G^j_k = \langle [r, d, v)]$ and $H^i_j = \langle [r', d', v']\rangle$. Then, $G; H = \langle [r'', d'', v'']\rangle$, for d'' the disjoint union of d and d', modulo the equivalence on nodes induced by $r(x) = v'(x)$ for all $x \in \underline{j}$, and $r'' : d_r \to d''$, $v'' : d_v \to d''$ the uniquely induced arrows. Let $G^i_j = \langle [r, d, v]\rangle$ and $H^k_l = \langle [r', d', v']\rangle$. Then, $G \oplus H = \langle [r'', d'', v'']\rangle$, for d'' the disjoint union of d and d', and $r'' : \underline{i + k} \to d'', v'' : \underline{j + l} \to d''$ the uniquely induced arrows.

4 Term Graph Inequational Deduction

As standard first-order terms denote derived operations in total algebras, so term graphs are particularly suited for denoting *derived relations* in a multi-algebra (see [2] for a detailed discussion about this).

Definition 4.1 (relation represented by a term graph) *Given a multi-algebra A over Σ and a term graph G of rank (j,i), by G^A we denote the relation $G^A : |A|^j \leftrightarrow |A|^i$ defined as follows. Let $\langle r,d,v\rangle$ be any ranked* DAG *in G, such that $N(d) = \{n_1,\dots,n_p\}$; then*

$$\langle b_1,\dots b_i\rangle \in G^A(\langle a_1,\dots a_j\rangle) \Leftrightarrow$$

$$\exists x_{n_1}\dots x_{n_p}\,.\left\{\begin{array}{l}\forall n\in N(d)\,.\,x_n\in|A|\ \wedge\\ \forall k\in \underline{j}\,.\,x_{v(k)}=a_k\ \wedge\\ \forall m\in N_\Sigma(d)\,.\,x_m\in l_d(m)^A(\langle x_{s_d(m)\uparrow 1},\dots,x_{s_d(m)\uparrow arity(l_d(m))}\rangle)\wedge\\ \forall h\in \underline{i}\,.\,x_{r(h)}=b_h\end{array}\right\}$$

wher $s_d(m)\uparrow i$ denotes the i-th element of the sequence $s_d(m)$, for each node $m\in N_\Sigma(d)$ and $i=1,\dots,arity(l_d(m))$.

In other words, we take an element of the carrier of the multi-algebra A for each node of the term graph. The tuples of elements corresponding to roots and variables of G are in relation G^A if each element corresponding to a node m such that $l_d(m)=f\in\Sigma_k$ belongs to $f^A(m_1,\dots,m_k)$, where $m_1,\dots,m_k$ are the successors of m. It is easy to prove that the definition is well-given, in the sense that it does not depend on the concrete representative chosen for G.

As an example, consider the term graph G_3 above. It has rank $(2,1)$, thus G_3^A is a function from $|A|^2$ to $\mathcal{P}(|A|)$, defined as follows:

$$b\in G_3^A(a_1,a_2)\Leftrightarrow$$

$$\exists x_1,x_2,x_g,x_f,x_h,x_k\in|A|\,.\,\{x_1=a_1\ \wedge\ x_2=a_2\ \wedge\ x_g\in g^A(x_2,x_1)\wedge$$
$$x_f\in f^A(x_g,x_2)\ \wedge\ x_h\in h^A(x_g,x_f)\ \wedge\ x_k\in k^A(x_2)\ \wedge\ b=x_h\}$$

where f^A, g^A, h^A and k^A are the fundamental relations corresponding to the interpretation of the operators of the signature in the multialgebra A.

As further examples, let us consider now two (quite degenerate) term graphs that will play some rôle in a moment. By $!_j$ (a *discharger*) we shall denote the only discrete term graph of rank $(j,0)$, thus having j variable nodes and no roots. By ∇_j (a *duplicator*) we shall denote the discrete term graph of rank $(j,2j)$ such that variable k is both the k-th and the $(k+j)$-th root, for $k\in\underline{j}$. By applying the general formula above, one easily gets (denoting $\langle\rangle$ the only element of $|A|^0$) that for all $a_1,\dots,a_j\in|A|$,

$$!_j^A(a_1,\dots,a_j)=\{\langle\rangle\},\text{ and}$$
$$\nabla_j^A(a_1,\dots,a_j)=\{\langle a_1,\dots,a_j,a_1,\dots,a_j\rangle\}$$

In particular, notice that both relations are total and univalent, that is, they are just functions.

Quite importantly, the mapping $(\_)^A$ from term graphs to derived relations over A can be easily shown to be a homomorphism with respect to the operations introduced in Definition 2.2 and Definition 3.4.

Lemma 4.2 *Let A be a multialgebra and F, G be two term graphs. Then $(F \oplus G)^A = F^A \oplus G^A$ and $(F\,;\,G)^A = F^A\,;\,G^A$.* □

Having chosen term graphs to denote derived relations in a multi-algebra, we can explore their use in specification. Following a well-established tradition in the theory of multi-algebras, we shall consider inequations, which are naturally interpreted as inclusions between the corresponding relations.

Definition 4.3 (term graph specifications) *A* (term graph) inequation over Σ *is a pair $\langle F, G\rangle$ of equally-ranked term graphs, usually written $F \sqsubseteq G$.*

Given a multi-algebra A over Σ, we say that inequation $F \sqsubseteq G$ holds in *A (and we write $A \models F \sqsubseteq G$) if $F^A \subseteq G^A$. Given a class $\mathcal{A}$ of multialgebras, the same inequation holds in $\mathcal{A}$ (written $\mathcal{A} \models F \sqsubseteq G$) if it holds for all $A \in \mathcal{A}$.*

A (term graph inequational) specification *is a pair $\langle \Sigma, \Phi\rangle$, where Φ is a set of term graph inequations over Σ. Given a specification $\langle \Sigma, \Phi\rangle$, the class of its* models *is* $\mathbf{MAlg}_{\Sigma,\Phi} = \{A \in \mathbf{MAlg}_\Sigma \mid A \models \varphi \text{ for all } \varphi \in \Phi\}$

In other words, an inequation $F \sqsubseteq G$ holds in A if relation F^A is included in relation G^A. The next step is to try and characterise syntactically the inclusion of derived relations in a multialgebra satisfying a given set of inequations.

Definition 4.4 (term graph inequational deduction) *Let $\langle \Sigma, \Phi\rangle$ be a term graph inequational specification. Then, an inequation $F \sqsubseteq G$ is deduced from the specification, written as $\Phi \vdash F \sqsubseteq G$, if the inequation can be obtained by the following set of axioms and inference rules*

[axiom]

$$\frac{\varphi \in \Phi}{\Phi \vdash \varphi}$$

[structural axioms] *For all term graphs G of rank (j, i),*

$$\frac{}{\Phi \vdash G; !_i \sqsubseteq !_j} \qquad \frac{}{\Phi \vdash G; \nabla_i \sqsubseteq \nabla_j; (G \oplus G)}$$

[reflexivity and transitivity] *For all G, H, K of rank (j, i),*

$$\frac{}{\Phi \vdash G \sqsubseteq G} \qquad \frac{\Phi \vdash G \sqsubseteq H, \quad \Phi \vdash H \sqsubseteq K}{\Phi \vdash G \sqsubseteq K}$$

[substitutivity] *For all G_1, H_1 of rank (j, i) and G_2, H_2 of rank (i, k),*

$$\frac{\Phi \vdash G_1 \sqsubseteq H_1, \quad \Phi \vdash G_2 \sqsubseteq H_2}{\Phi \vdash G_1; G_2 \sqsubseteq H_1; H_2}$$

[congruence] *For all G_1, H_1 of rank (j,i) and G_2, H_2 of rank (k,l),*

$$\frac{\Phi \vdash G_1 \sqsubseteq H_1, \quad \Phi \vdash G_2 \sqsubseteq H_2}{\Phi \vdash G_1 \oplus G_2 \sqsubseteq H_1 \oplus H_2}$$

Let us comment on the structural axioms. As discussed above, for every multi-algebra A the interpretation of $!_j^A$ is fixed, and it is the total function from $|A|^j$ to the singleton set $|A|^0$. Therefore, whatever is the relation $G^A : |A|^j \leftrightarrow |A|^i$, the composition $G^A ; !_i^A$ is a possibly partial function to a singleton set, thus $(G ; !_i)^A \subseteq !_j^A$ necessarily holds.

Concerning the second structural axiom, it formalises the fact that if we take twice (using the duplicator) the result of evaluating the non-deterministic expression represented by G, in general we obtain less results than evaluating it twice, because the two evaluations may deliver different results (see [3] for more details about this).

Note also that the closure with respect to the operations explicitly ensures that substitutivity holds. Given this explanation of the structural axioms and Lemma 4.2, it comes as no surprise that the above calculus is sound.

Theorem 4.5 (soundness of term graph inequational deduction) *Let $\langle \Sigma, \Phi \rangle$ be an inequational specification, and $F \sqsubseteq G$ an inequation over Σ:*

$$\Phi \vdash F \sqsubseteq G \quad \Longrightarrow \quad \mathbf{MAlg}_{\Sigma,\Phi} \models F \sqsubseteq G$$

□

We were not able to prove the completeness of the calculus. We conjecture it is indeed so, as far as suitable restrictions on the specifications are imposed (see also Definition 5.1).

We conclude this section by showing some examples of inequations, and describing the induced properties on the corresponding derived relations.

Proposition 4.6 (specifying with term graph inequations) *Let A be a Σ-multi-algebra, and G a term graph of rank (j,i). Then*

- $A \models !_j \sqsubseteq G ; !_i$ *iff G^A is a total relation.*
- $A \models \nabla_j ; (G \oplus G) \sqsubseteq G ; \nabla_i$ *iff G^A is univalent.*

Furthermore, let us consider the term graph H of rank (j,k). Then, the derived operator $F = \nabla_j ; (G \oplus (H ; !_k))$ represents a domain restriction *of G wrt. the domain of definition of H, that is, for all multialgebras A, both $F^A \subseteq G^A$ and $dom(F^A) = dom(H^A) \cap dom(G^A)$ hold. Then, we have*

- $A \models \nabla_j ; (G \oplus (H ; !_k)) \sqsubseteq G$.
- $A \models G \sqsubseteq \nabla_j ; (G \oplus (H ; !_k))$ *iff $dom(G^A) \subseteq dom(H^A)$.* □

5 Term Graph Rewriting for Inequational Deduction

In this section we show how to transform a term graph specification into a term graph rewriting system, in such a way that an inequation $F \sqsubseteq G$ can be deduced

using the system of Definition 4.4 if, and only if, F rewrites to G according to the associated rewriting system.

The relationship between rewriting and logical deduction that we are going to describe follows the same pattern of the relationship between term rewriting and rewriting logic, which is "folklore", and that is extended to some generalizations of rewriting and logics for example in [7].

We shall consider term graph inequations with a restricted format. Firstly, according to Definition 4.3, both sides of an inequation must have the same rank. Secondly, rules must be *proper*, a condition generalising the requirement, usual in term rewriting, that a rule cannot have a variable as left-hand side.

Definition 5.1 (proper inequations) *A term graph F is* proper *if no variable node in F is also a root; an inequation $F \sqsubseteq G$ is proper if F is.*

Now, we can describe our encoding.

Definition 5.2 (from inequations to rules) *Given a term graph F, we shall denote by $|F|$ an arbitrarily chosen, unranked* DAG *belonging to F. Let $F \sqsubseteq G$ be a proper inequation, both of whose term graphs have rank (j, i). The associated rule $R_G^F = \left(d_L \xleftarrow{l} d_K \xrightarrow{r} d_R\right)$ is defined as follows:*

- *d_L and d_R are the unranked* DAG*'s $|F|$ and $|G|$, respectively;*
- *the interface d_K is a discrete graph containing $i + j$ nodes;*
- *the morphisms l and r map the $i + j$ nodes of d_K consistently to the i roots and the j variables of $d_L = |F|$ and $d_R = |G|$, respectively.* [7]

Example 5.3 The deduction system of the previous section includes the (structural) inequations $F;! \sqsubseteq !$ and $F;\nabla \sqsubseteq \nabla;(F \oplus F)$. Assuming that F is a term graph including only one root labeled by a k-ary operator $f \in \Sigma$ and k distinct variables, the two corresponding rules, that we denote $R_f^!$ and R_f^∇, respectively, have the shape depicted in Figure 2 and Figure 3.

The DAG on the left-hand side (center, right-hand side) is d_L (d_K and d_R, respectively). The x_i, y_j's indicate empty nodes, and are used to intuitively describe the span of DAG morphisms: as an example, in rule R_f^∇ the nodes identified by y_{f1} and y_{f2}, distinct in d_K, are coalesced into f in d_L.

Note that the left morphism l might not be injective: this happens in the encoding of a proper inequation $F \sqsubseteq G$ exactly when two roots in F coincide (like in rule R_f^∇).

[7] Informally, we assume here that we kept track of which nodes of $|F|$ ($|G|$) were roots or variables in F (G, respectively).

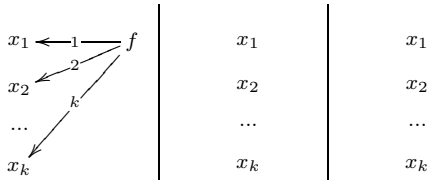


Figure 2. The rewriting rule $R^!_f$ encoding inequation $G_f; ! \sqsubseteq !$.

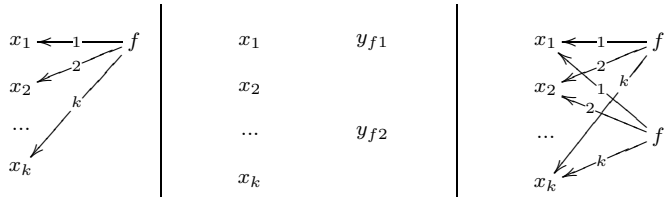


Figure 3. The rewriting rule R^∇_f encoding inequation $G_f; \nabla \sqsubseteq \nabla; (G_f \oplus G_f)$.

The next lemma summarises some results concerning the existence of relevant pushouts and pushout complements, needed to perform the DPO construction in **DAG** using rules in the format described above. We shall consider plain morphisms only as matches (see Definition 3.1) in view of Lemma 7.3.

Lemma 5.4 (pushout complements in DAG) *Let $l : d_K \to d_L$ and $r : d_K \to d_R$ be as in Definition 5.2, and let $m : d_L \to d_G$ be a plain match which satisfies the dangling condition.*[8] *Then:*

(i) *If l is injective, then there exists a lattice of pushout complements of l and m, ordered by existence of a commuting morphism. If $d_K \xrightarrow{k} d_D \xrightarrow{d} d_G$ is the minimal pushout complement,*[9] *then the pushout of $k : d_K \to d_D$ and $r : d_K \to d_R$ exists.*

(ii) *If l is not injective, then there is a non-empty family of lattices of pushout complements, and each of these lattices enjoys the same property as in the previous point.* □

Since term graph rewriting has been defined over unranked DAG's, but term graph inequations are ranked, we have the need to "encode" in some way the rank inside the structure of a simple DAG.

Definition 5.5 (rewriting of ranked term graphs) *For each $k \in \mathbb{N}$, let $\hat{k}$ be an operator of arity k not in Σ, and let $G_{\hat{k}}$ be the term graph of rank $(k, 1)$ having one non-variable node labeled by $\hat{k}$ and k variable nodes.*

Let F, H be term graphs (over Σ) of rank (j, k). We say that F rewrites to H if $|\hat{F}| \Rightarrow^*_{\mathcal{R}} |\hat{H}|$, where $\hat{F} \stackrel{def}{=} F; G_{\hat{k}}$.

[8] This is a standard application condition for DPO approach: formally, for each node $n \in N(d_L) \backslash l(N(d_K))$, if $m(n) \in s_{d_G}(n')$ then $n' \in m(N(d_L))$. The reader familiar with the DPO approach may note that the identification condition is ensured by the Σ-injectivity of m.

[9] It is still open if this pushout complement can be characterized as the "natural" one, i.e., being a pullback, as in [5].

Note that $|\hat{F}| \Rightarrow^*_{\mathcal{R}} |\hat{G}|$ implies $|F| \Rightarrow^*_{\mathcal{R}} |G|$, but the converse does not hold, because of the dangling condition. We are now ready to state the main result.

Theorem 5.6 (inequational deduction as rewriting) *Let $\langle \Sigma, \Phi \rangle$ be an inequational specification, and let $\mathcal{R}(\Phi)$ be the set of rules including the structural rules $R^!_f$ and R^∇_f for all $f \in \Sigma$ (see Example 5.3), and one rule for each inequation $\varphi \in \Phi$ as for Definition 5.2. Then*

$$\Phi \vdash F \sqsubseteq G \qquad \textit{iff} \qquad |\hat{F}| \Rightarrow^*_{\mathcal{R}(\Phi)} |\hat{G}|$$

□

A sketch of the proof can be found in the Appendix.

6 Conclusions and Further Works

The main goal of our paper was to find evidences for supporting the claim that term graphs should play for the specification of multi-algebras the same rôle that standard terms play for total algebras. The technical justification of this comes from a functorial presentation of multi-algebras, originally proposed in [2], where term graphs are proved to be a suitable syntactical tool for the presentation of derived operators in multialgebras.

Our claim was substantiated by presenting a calculus for term graph inequational deduction, and proving its soundness. In particular, the calculus takes full advantage of the use of term graphs, because an unrestricted substitutivity rule holds (which does not hold for other calculi for multi-algebras). More importantly, we proved that the calculus can be implemented by means of term graph rewriting, in such a way that rewriting steps mimic inequational deduction.

As future work, the first problem we would like to tackle is the completeness of the calculus. At the same time, we would also like to compare our calculus with other proposals from the literature with respect to both the complexity of rules, and the expressive power.

Further extensions we foresee include Horn clause logic, and some limited form of negation which appears to be needed in order to have the expressive power to specify satisfactorily, for example, a non-deterministic choice operator (see, e.g., [11]).

References

[1] A. Corradini and F. Gadducci. An algebraic presentation of term graphs, via gs-monoidal categories. *Applied Categorical Structures*, 7:299–331, 1999.

[2] A. Corradini and F. Gadducci. A functorial semantics for multi-algebras and partial algebras, with applications to syntax. *Theoret. Comput. Sci.*, 2002. To appear. Available at http://www.di.unipi.it/~gadducci/papers.

[3] A. Corradini, F. Gadducci, and W. Kahl. Term graph syntax for multi-algebras. Technical Report TR-00-04, University of Pisa, Department of Informatics, 2000.

[4] A. Corradini and F. Rossi. Hyperedge replacement jungle rewriting for term rewriting systems and logic programming. *Theoret. Comput. Sci.*, 109:7–48, 1993.

[5] H. Ehrig and H.-J. Kreowski. Categorical approach to graphic systems and graph grammars. In G. Marchesini and S.K. Mitter, editors, *Mathematical Systems Theory*, volume 131 of *Lecture Notes in Economics and Mathematical Systems*, pages 323–351. Springer, 1976.

[6] H. Ehrig, M. Pfender, and H.J. Schneider. Graph-grammars: an algebraic approach. In R.V. Book, editor, *Switching and Automata Theory*, pages 167–180. IEEE Computer Society Press, 1973.

[7] F. Gadducci. *On the Algebraic Approach to Concurrent Term Rewriting*. PhD thesis, University of Pisa, Department of Informatics, 1996.

[8] D. Plump. Term graph rewriting. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 2, pages 3–61. World Scientific, 1999.

[9] M.R. Sleep, M.J. Plasmeijer, and M.C.J.D. van Eekelen, editors. *Term Graph Rewriting: Theory and Practice*. Wiley, 1993.

[10] M. Walicki and S. Meldal. Algebraic approaches to nondeterminism: An overview. *ACM Computing Surveys*, 29:30–81, 1997.

[11] M. Walicki and S. Meldal. A complete calculus for the multialgebraic and functional semantics of nondeterminism. *ACM Trans. Program. Lang. Syst.*, 17:366–393, 1997.

7 Appendix

We present here some proof sketches of the main results of the paper. We begin with a technical lemma.

Lemma 7.1 (pushouts in DAG) *Let $f : d \to d'$ be a* DAG *morphism, and let its associated* substitution *be the function $\varphi_f : N_\emptyset(d) \to T_\Sigma(N_\emptyset(d'))$, mapping each variable node n to the term over $(\Sigma, N_\emptyset(d'))$ obtained by "unraveling" $f(n)$.*

Given a span $d' \xleftarrow{f} d \xrightarrow{g} d''$ in **DAG**, *its pushout exists if and only if substitutions φ_f and φ_g unify.* □

For a proof, see Proposition 1.18 of [4], where the statement is proved for category $\mathbf{Jungle}_\Sigma$, which is equivalent to category **DAG**. We can now provide a proof skecth of Lemma 5.4.

Proof sketch of Lemma 5.4. For point (i), informally, the minimal pushout complement object d_D is obtained from d_G by deleting all nodes in $m(N(d_L)) \backslash l(N(d_K))$, and by making the labeling and successor functions undefined on the nodes in $m(l(N(d_K)) \backslash N_\emptyset(d_L))$. Given two pushout complements $d_K \xrightarrow{k'} d'_D$ and $d_K \xrightarrow{k''} d''_D$, their join is their pushout, and their meet is determined by the pullback of the injections in the pushout.

The existence of the right pushout of Figure 1 for the minimal pushout complement follows from the previous lemma, by observing that out of the $j + i$ nodes of d_K, j are mapped by r to distinct variables of d_R (by construction), and the other i are mapped by k to distinct variables of d_D, thus the corresponding substitutions unify.

For point (ii), intuitively, if $l(n) = l(n')$, then in a pushout complement object the references to node $m(l(n))$ in d_G can be split in an arbitrary way as references to $k(n)$ or to $k(n')$, and each such a choice determines a lattice of pushout complements. □

The rest of the section is devoted to sketch the proof of our main result, Theorem 5.6. We first start with an auxiliary definition.

Definition 7.2 (term graph contexts) *A* (term graph) context $C[j,i]$ *is a well-formed, ranked term graph expression (using term graphs as constants, and the operators* $\oplus$ *and* $;$*) containing exactly one (ranked) place-holder* $[j,i]$ *(for some* $i,j \in \mathbb{N}$*). If* $C[j,i]$ *is a context and* G *is a term graph of rank* (j,i)*, then by* $C[G]$ *we denote the term graph obtained by evaluating the expression* C *after replacing* $[i,j]$ *by* G*.*

Now, we have two technical lemmas. The first relates the connection between plain matches satisfying the dangling condition and contexts; building on that, the second, fundamental lemma states that any rewriting step obtained by applying a rule R is equivalent to embedding R in a context.

Lemma 7.3 (plain morphisms and contexts) *Let* F *be a term graph of rank* (j,i)*, let* $d_K \xrightarrow{l} d_L$ *be the left component of a rule* R_H^F *(as for Definition 5.2) and let* $d_G = |G|$ *be a chosen* DAG *in a term graph* G*. Then there is a context* $C[j,i]$ *such that* $G = C[F]$ *if and only if there is a plain morphism* $m : d_L \to d_G$ *satisfying the dangling condition with respect to* l*.* □

Lemma 7.4 (rewriting as contextualization) *Let* R_G^F *be the rule associated with the proper inequation* $F \sqsubseteq G$*, and let* S*,* T *be two term graphs. Then* $|\hat{S}| \Rightarrow_{R_G^F} |\hat{T}|$ *if and only if there exists a context* C *such that* $S = C[F]$ *and* $T = C[G]$*.* □

Finally, we can provide the outline of the proof the main result.

Proof sketch of Theorem 5.6. **[Deduction implies rewriting]** If $\Phi \vdash F \sqsubseteq G$, we show that $|\hat{F}| \Rightarrow^*_{\mathcal{R}(\Phi)} |\hat{G}|$ by induction on the last rule used for the deduction. For structural axioms, their effect in deduction can be simulated by repeated applications of the auxiliary rules of Example 5.3. The reflexivity and transitivity rules are handled in the obvious way. For substitutivity and congruence, we exploit the last lemma. If $\Phi \vdash G_1 \sqsubseteq H_1$ and $\Phi \vdash G_2 \sqsubseteq H_2$, by induction hypothesis we have $|\hat{G_1}| \Rightarrow^*_{\mathcal{R}(\Phi)} |\hat{H_1}|$ and $|\hat{G_2}| \Rightarrow^*_{\mathcal{R}(\Phi)} |\hat{H_2}|$. From this it follows that $|G_1| \Rightarrow^*_{\mathcal{R}(\Phi)} |H_1|$. Consider now the context $C = G_1; [j,i]$, where (j,i) is the rank of G_2. By Lemma 7.4, and by induction on the length of $|\hat{G_2}| \Rightarrow^*_{\mathcal{R}(\Phi)} |\hat{H_2}|$, we obtain that $|C[\hat{G_2}]| \Rightarrow_{\mathcal{R}(\Phi)} |C[\hat{H_2}]|$, thus $|G_1 \hat{;} G_2| \Rightarrow_{\mathcal{R}(\Phi)} |G_1 \hat{;} H_2|$. Acting symmetrically with context $[k,j]; H_2$, where (k,j) is the rank of G_1, one obtains the desired rewriting sequence. Similarly for $\oplus$.

[Rewriting implies deduction] One first shows that a single rewriting step corresponds to a deduction. This is trivial for steps corresponding to the rules of $R(\Phi)$. For a general rewriting, by Lemma 7.4 this is equivalent to putting the rule in a term graph context. The proof then goes by induction on the syntactical structure of the context, observing that term graphs used as constants correspond to the reflexivity rule, while operations $\oplus$ and $;$ correspond to the application of the congruence and substitutivity rules. Next, rewriting sequences consisting of more

than one step correspond to the application of the transitivity rule. □