A Complete Proof System for Nested Term Graphs

Stefan Blom

Department of Computer Science, Vrije Universiteit Amsterdam,
de Boelelaan 1081, 1018 HV Amsterdam, The Netherlands

Abstract. Nested Term Graphs are syntactic representations of cyclic term graphs. Via a simple translation they contain μ-terms as a subset. There exists a characterization of the μ-terms that unwind to the same tree, presented as a complete proof system. This paper gives a similar characterization for Nested Term Graphs. The semantics of tree unwinding is presented via bisimulations.

1 Introduction

Sharing and cycles are essential features in a program development system [10]. In fact, they occur at the source level, after parsing, in the intermediate program representation and during program execution. However, traditional computational models such as term rewriting systems [8] and λ-calculus [3] do not allow to reason about them directly. A natural way to capture sharing and cycles is to rewrite graphs instead of terms. Since the graphs considered here have locally the same structure as terms, we will call them term graphs as is nowadays common practice [11]. A term graph can be represented as sets of nodes and edges [4, 12, 5], as terms with loop-back operators [7, 13, 6] or as sets of recursive equations [1]. In this paper we will follow the latter approach.

Actually we will consider not only 'flat' systems of recursion equations, but also allow nesting of the recursion construct in the same way as it was done in [2] for the λ-calculus. This is inspired by the simple and elegant μ-calculus in which many recursive behaviors can be expressed. Clearly the μ-calculus can also be viewed as a simple form of a calculus for recursion expressions where nesting is present. However the μ-calculus is not as expressive as one would wish, since it does not allow 'subterm sharing' (In the terminology of [1], the μ-calculus admits 'vertical sharing', but not 'horizontal sharing'.) On the other hand there is a simple complete proof system for the μ-calculus [1], where complete refers to the semantics of μ-terms obtained by infinite tree unwinding, or what is equivalent, to bisimilarity. This proof system is very reminiscent of the complete proof system in [9] for recursive expressions in the area of communicating processes. In the present paper we generalize μ-terms to our nested term graphs (NTG's) and also construct a complete proof system for these NTG's. Here the semantics is again given by bisimilarity.

Historically, recursive or iterative expressions have been studied widely and deeply [7, 13, 6]. There are two major points of difference between this 'classical' work and the present paper. First, the μ-calculus and the present generalized system of NTG's employ the feature of bound variables as in λ-calculus. This has both advantages and disadvantages; a disadvantage is the need for a careful treatment of renaming just as in λ-calculus, an advantage is the greater ease of expressing what one wants. The same situation arises in λ-calculus versus the variable-free system of combinatory logic, and we will refrain from further discussing this issue here. An advantage of the approach in [7, 13, 6] is that the treatment can be more algebraic since there are only first order operators like Kleene star or, dagger. By contrast, our treatment can be called 'higher-order', where this phrase is meant to indicate the presence of bound variables, thus going beyond the first-order framework. Another reason for nesting is, that nesting allows application of term rewriting on a same syntactic level, just as the μ-calculus does. After a minor syntactic change the framework of [7, 13, 6] allows for this too, but the way pointers are represented causes a lot of very small very simple steps when one has to unwind the graph to expose a redex. While the notion of term graph rewriting in [4, 12, 5] has more expressive power in general than term rewriting, term rewriting is easier to reason about and the term graph rewriting systems of main interest to this line of research can be expressed as simple extensions of term rewriting systems. The second major difference between our work and that in [7, 13, 6] is the following. As semantics for term graphs and μ-terms infinite tree unwinding is used. In [1] it was shown that bisimulation gives the same equivalence as infinite tree unwinding. The work on terms with loop-back operators uses bisimulation as semantics when soundness is considered, but in many cases proves completeness only with respect to graph isomorphism. Completeness with respect to bisimulation is proven for restricted versions of the theory. These restricted versions are applicable to term graphs, but only after encoding of the problem.

There are many NTG's representing the same graph and many graphs with the same semantics. The natural question to ask is: which NTG's have the same semantics? This paper answers the question in the form of a complete proof system. We will use bisimulation instead of unwinding as the basic notion of equality on NTG's. Finally, we want to remark that our work can be seen as a first step in developing the proof theory of the fundamental programming constructs let and letrec.

The rest of the paper is organized as follows: In section 2 we introduce the notion of nested term graph and related notions such as substitution on NTG's. In section 3 we introduce the notion of equality on NTG's: bisimulation. Actually bisimulation is defined on term graphs so we must translate NTG's to term graphs. The translation is called the underlying graph. After this we are ready to specify the proof system in section 4 and prove its soundness (section 5) and completeness (section 6).

2 Syntax

We will adopt the convention that $\mathcal{V}$ is a set of variables with typical members α, β, γ, δ, α_1, α_2, ... and that $\mathcal{F}$ is a set of function symbols $f, g, h, F, G, H, \ldots$ each with a given fixed arity. By arity(f) we mean the arity of the function symbol f, constants are function symbols with arity 0. There is a special constant $\bullet$, the *black hole*.

Definition 1. A *nested term graph* can be:

- α, for each $\alpha \in \mathcal{V}$,
- $f(t_1, \ldots, t_n)$ for each n-ary $f \in \mathcal{F}$ and NTG's $t_1, \ldots, t_n$ or
- $\langle s | \alpha_1 = t_1, \ldots, \alpha_n = t_n \rangle$, for NTG's $s, t_1, \ldots, t_n$ and variables $\alpha_1, \ldots, \alpha_n$ which are pairwise different.

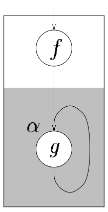

Fig. 1. The picture of $\langle f(\alpha) | \alpha = g(\alpha) \rangle$.

We will now introduce some terminology and notation. Some of the terminology has its roots in the pictures we can draw of NTG's. An example of such a picture is given in Fig. 1.

- Instead of $\langle s | E \rangle$ we will often write s^E.
- Given a nested term graph of the form $\langle s | \alpha_1 = t_1, \ldots, \alpha_n = t_n \rangle$, we refer to the entire NTG as a *box*, to the s as the *external part*, and to the set of equations $\alpha_1 = t_1, \ldots, \alpha_n = t_n$ as the *internal part* or the *environment*. The symbols $E, F, \ldots$ are normally used to match environments.
- A NTG without box constructs is a *term*.
- A nested term graph is *guarded* if it is of the form $f(t_1, \ldots, t_n)$.
- A nested term graph of the form α^E where all right-hand sides of the equations in E are of the form $f(\alpha_1, \ldots, \alpha_n)$ is called *flat*.
- We say that an occurrence of a variable is a *free occurrence*, if the occurrence is not inside a box, where the variable occurs as the left-hand side of an equation in the environment. If a variable occurs as the left-hand side of an equation in the internal part, then that occurrence is a *binding occurrence* and all free occurrences of the variable in the external part and in the right-hand sides of the environment are *bound* by this left-hand side occurrence.

– For any nested term graph x or environment x we define $\mathrm{var}(x)$ to be the set of variables occurring in x and $\mathrm{free}(x)$ to be the set of variables having at least one free occurrence in x. We define $\mathrm{def}(\{\alpha_1 = t_1, \ldots, \alpha_n = t_n\})$ to be the set of variables the environment defines: $\{\alpha_1, \ldots, \alpha_n\}$.

Example 1. (free and bound variables)

$$\langle f(\alpha_1, \langle f(\alpha_2, \beta_1) | \alpha_3 = f(\alpha_4, \gamma), \beta_2 = \bullet \rangle) | \alpha_5 = f(\alpha_6, \beta_3) \rangle$$

Here the subscripts denote the number of the occurrence. The only occurrence of γ is a free occurrence, the 1st occurrence of α is bound to the 5^{th}occurrence, the $2^{\text{nd}}$and 4^{th}occurrences of α are bound to the 3^{rd}occurrence, the 1^{st}occurrence of β is bound to the $2^{\text{nd}}$and the 3^{rd}occurrence of β is free.

Definition 2. When t is obtained from s by replacing in s a binding occurrence of a variable and all occurrences bound to that occurrence by a variable that does not occur is s, we say that $s =_\alpha t$. We denote by the same $=_\alpha$ the transitive reflexive closure of the relation and we name it *α-conversion*. A nested term graph is maximally α-converted if no variable occurs both free and as a binder and no variable occurs twice as a binder. Unless stated otherwise we will not distinguish between two α-convertible nested term graphs.

Definition 3. A context $C[.]$ is a NTG with one occurrence of a constant replaced by $\Box$. By $C[s]$ we mean $C[.]$ with the box replaced by s.

Example 2. Following the definition these are contexts:

$$\langle \Box | \alpha = f(\alpha) \rangle \quad \langle \alpha | \alpha = f(\Box) \rangle \quad f(\Box)$$

and these are not:

$$\langle \Box | \alpha = f(\Box) \rangle \quad \langle \alpha | \Box = f(\alpha) \rangle$$

Definition 4. A *substitution* is a partial function σ from $\mathcal{V}$ to the set of NTG's. If the NTG the substitution is applied to satisfies the condition that the set of variables occurring in the left-hand sides in the NTG is disjoint from the set of variables the substitution is defined for and from the set of variables occurring free in the images of variables under the substitution then the substitution extends to NTG's by the following rules:

$$\begin{aligned} \alpha\sigma &= \begin{cases} \sigma(\alpha) & \text{, if } \alpha \in D_\sigma \\ \alpha & \text{, otherwise} \end{cases} \\ f(t_1, \ldots, t_n)\sigma &= f(t_1\sigma, \ldots, t_n\sigma) \\ \langle s | \alpha_1 = t_1, \ldots, \alpha_n = t_n \rangle\sigma &= \langle s\sigma | \alpha_1 = t_1\sigma, \ldots, \alpha_n = t_n\sigma \rangle \ . \end{aligned}$$

Because NTG's are taken modulo α-conversion the condition can always be satisfied.

3 Bisimulation

We want to consider NTG's equivalent if they unwind to the same rational tree. On term graphs there is the equivalent notion of bisimilarity ([1]). This notion is useful because it is decidable and it will give us the strategy for the completeness proof later on. To define bisimilarity on NTG's we first associate a term graph with every NTG. This term graph, called the underlying graph, forgets about boxes. The underlying graph is defined using an i-successor relation on occurrences of function and variable symbols.

Definition 5. We define i-successor on occurrences of function and variable symbols as follows:

- $x \rightarrow_0 y$ if x is the left-hand side of an equation and y is the head symbol of the same equation.
- $x \rightarrow_0 y$ if x is a bound variable and y is the left-hand side of the equation x is bound to.
- $x \rightarrow_0 x$ if x is an occurrence of the black hole $\bullet$.

We put $x \sim_0 y$ if x and y are free occurrences of the same variable.

The relation $\sim$ on occurrences of both variables and function symbols is the equivalence relation generated by the union of $\rightarrow_0$ and $\sim_0$.

We say $x \rightarrow_i y$ $(i > 0)$ if x is the occurrence of a function symbol and y is the head symbol of the i^{th} argument of that symbol. We denote the equivalence class that a symbol x belongs to by $\overline{x}$.

The head symbol is the same as the left most symbol of any NTG and is denoted by $\mathrm{lm}(t)$ for any NTG t. The head symbol of a sub NTG t of $C[t]$ is denoted by $\mathrm{lm}(t \subset C[t])$.

Definition 6. The *underlying graph* of a nested term graph t , notation $\mathrm{graph}(t)$, is a term graph.

The nodes are the equivalence classes of the equivalence relation $\sim$. The class a symbol x belongs to is denoted by $\overline{x}$. If a class contains an occurrence of a function symbol that function symbol is the label. If a class contains an occurrence of a free variable that variable is the label. Otherwise $\bullet$ is the label.

If $x \rightarrow_i y$ $(i > 0)$ then $\overline{y}$ is the i^{th} argument of $\overline{x}$. We will denote this by $\overline{x} \rightarrow_i \overline{y}$.

The relations $\rightarrow_i$ $(i > 0)$ are now defined on both occurrences of symbols in a NTG and on nodes of the underlying graphs. This overloading is harmless. Note that $\rightarrow_0$ is only defined on occurrences. The underlying graph is actually a term graph.

Example 3. Figure 2 illustrates the $\rightarrow_i$ relations and the equivalence classes in the construction of the underlying graph of $\langle f(\alpha) | \alpha = g(\langle \alpha | \alpha = f(\beta) \rangle), \beta = f(\alpha) \rangle$. The resulting underlying graph is drawn in Fig. 3.

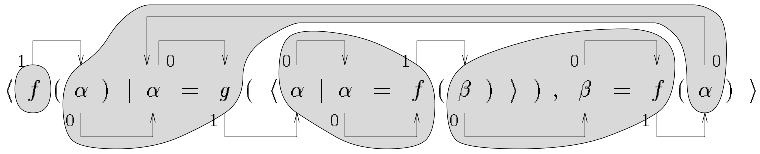

Fig. 2. The construction of an underlying graph

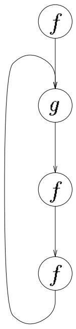

Fig. 3. An example of an underlying graph

Definition 7. For two graphs g and h a relation $R \subset \text{nodes}(g) \times \text{nodes}(h)$ is a *bisimulation* if

$$(\text{root}(g), \text{root}(h)) \in R$$

and

$$\forall (a, b) \in R : \text{label}(a) = \text{label}(b) \wedge \forall a', b', i : (a \rightarrow_i a' \wedge b \rightarrow_i b') \Rightarrow ((a', b') \in R) \ \ .$$

If such a relation exists we say g and h are *bisimilar* ($g \underline{\leftrightarrow} h$). We say that two NTG's s and t are *bisimilar*, denoted $s \underline{\leftrightarrow} t$ if $\text{graph}(s) \underline{\leftrightarrow} \text{graph}(t)$.

4 The Proof System

We assume that the expressions occurring in the proof system are maximally α-converted, i.e. all variables occurring as left-hand sides are pairwise different, also if they occur in different environments, and there is no overlap between free and bound variables. The equations we will reason about in the proof system will be written $s =_{\mathsf{p}} t$ to distinguish the equal sign used from the equal sign in the syntax of NTG's. If there is no confusion we will sometimes write $s = t$.

Some remarks now on how to read the system. Reflexivity, symmetry and transitivity are obvious. The black hole rule is there to express that defining $\alpha = \alpha$ is the same as explicitly saying it is undefined. Since the undefinedness

Table 1. the proof system

A1	$t =_{\mathsf{P}} t$	*reflexivity*
A2	$\langle s \mid \alpha = \alpha, E\rangle =_{\mathsf{P}} \langle s \mid \alpha = \bullet, E\rangle$	*black hole*
A3	$C[\alpha] =_{\mathsf{P}} C[t]$, if $C[\alpha] \equiv \langle s \mid E\rangle$ with $\alpha = t \in E$	*substitution*
A4	$\langle s \mid \rangle =_{\mathsf{P}} s$	*box removal*
A5	$f(s_1, \ldots, s_n)^E =_{\mathsf{P}} f(s_1^E, \ldots, s_n^E)$	*unwinding*
A6	$s^{E,F} =_{\mathsf{P}} s^E$, if $\mathrm{def}(F) \cap \mathrm{free}(s^E) = \emptyset$	*garbage collection*
R1	$\dfrac{s =_{\mathsf{P}} t}{t =_{\mathsf{P}} s}$	*symmetry*
R2	$\dfrac{s =_{\mathsf{P}} t \quad t =_{\mathsf{P}} u}{s =_{\mathsf{P}} u}$	*transitivity*
R3	$\dfrac{s =_{\mathsf{P}} t}{C[s] =_{\mathsf{P}} C[t]}$	*context*
R4	$\dfrac{\begin{array}{c} s_1 =_{\mathsf{P}} t_1\{s_1/\alpha_1, \ldots, s_n/\alpha_n\} \\ \vdots \\ s_n =_{\mathsf{P}} t_n\{s_1/\alpha_1, \ldots, s_n/\alpha_n\} \end{array}}{s_1 =_{\mathsf{P}} \langle \alpha_1 \mid \alpha_1 = t_1, \ldots, \alpha_n = t_n\rangle}$, for guarded $t_1, \ldots, t_n$	*folding*

is of a special kind we use a special symbol for it $\bullet$. Substitution requires some care. We don't want e.g.

$$\langle\langle \alpha \mid \beta = A\rangle \mid \alpha = F(\beta)\rangle =_{\mathsf{P}} \langle\langle F(\beta) \mid \beta = A\rangle \mid \alpha = F(\beta)\rangle$$

to be derivable as it is clearly wrong. The cause of the trouble is that there are both free and bound occurrences of β. The maximal α-conversion convention saves us here. Box removal and garbage collection are used to get rid of old unused definitions. Unwinding or distribution expresses the equivalence of shared definitions and identical local definitions. The context rule needs to be read carefully. Maximal α-conversion doesn't mean that variables that occur free in s or t in the premise are still free in the conclusion. I.e. deriving

$$\langle \alpha \mid \alpha = A\rangle =_{\mathsf{P}} \langle\langle \alpha \mid \rangle \mid \alpha = A\rangle$$

from

$$\alpha =_{\mathsf{P}} \langle \alpha \mid \rangle$$

is legal. The context rule is used to replace any subterm by an equal one. The folding rule states that a system of guarded recursive equations has a solution of a special form.

Definition 8. We say that s and t are semantically equal iff s and t are bisimilar:

$$\models s =_{\mathsf{P}} t \iff s \leftrightarroweq t\ .$$

Table 2. The formal proof from Ex. 4

1	$\langle f(\alpha)\vert\alpha = \langle\beta\vert\beta = f(\alpha)\rangle\rangle =_{\mathsf{P}} f(\langle\alpha\vert\alpha = \langle\beta\vert\beta = f(\alpha)\rangle\rangle)$	unwinding
2	$\langle\alpha\vert\alpha = \langle\beta\vert\beta = f(\alpha)\rangle\rangle =_{\mathsf{P}} \langle\langle\beta\vert\beta = f(\alpha)\rangle\vert\alpha = \langle\beta\vert\beta = f(\alpha)\rangle\rangle$	substitution
3	$\langle\beta\vert\beta = f(\alpha)\rangle =_{\mathsf{P}} \langle f(\alpha)\vert\beta = f(\alpha)\rangle$	substitution
4	$\langle f(\alpha)\vert\beta = f(\alpha)\rangle =_{\mathsf{P}} \langle f(\alpha)\vert\rangle$	garbage collection
5	$\langle f(\alpha)\vert\rangle =_{\mathsf{P}} f(\alpha)$	box removal
6	$\langle\beta\vert\beta = f(\alpha)\rangle =_{\mathsf{P}} f(\alpha)$	transitivity, 3, 4, 5
7	$\langle\langle\beta\vert\beta = f(\alpha)\rangle\vert\alpha = \langle\beta\vert\beta = f(\alpha)\rangle\rangle =_{\mathsf{P}} \langle f(\alpha)\vert\alpha = \langle\beta\vert\beta = f(\alpha)\rangle\rangle$	context, 6
8	$\langle f(\alpha)\vert\alpha = \langle\beta\vert\beta = f(\alpha)\rangle\rangle =_{\mathsf{P}} f(\langle\alpha\vert\alpha = \langle\beta\vert\beta = f(\alpha)\rangle\rangle)$	unwinding
9	$\langle\alpha\vert\alpha = \langle\beta\vert\beta = f(\alpha)\rangle\rangle =_{\mathsf{P}} f(\langle\alpha\vert\alpha = \langle\beta\vert\beta = f(\alpha)\rangle\rangle)$	transitivity, 2, 7, 8
10	$\langle\alpha\vert\alpha = \langle\beta\vert\beta = f(\alpha)\rangle\rangle =_{\mathsf{P}} \langle\delta\vert\delta = f(\delta)\rangle$	folding, 9
11	$f(\langle\alpha\vert\alpha = \langle\beta\vert\beta = f(\alpha)\rangle\rangle) =_{\mathsf{P}} f(\langle\delta\vert\delta = f(\delta)\rangle)$	context, 10
12	$\langle f(\alpha)\vert\alpha = \langle\beta\vert\beta = f(\alpha)\rangle\rangle =_{\mathsf{P}} f(\langle\delta\vert\delta = f(\delta)\rangle)$	transitivity, 1, 11
13	$\langle\alpha\vert\alpha = f(\alpha)\rangle =_{\mathsf{P}} \langle f(\alpha)\vert\alpha = f(\alpha)\rangle$	substitution
14	$\langle f(\alpha)\vert\alpha = f(\alpha)\rangle =_{\mathsf{P}} f(\langle\alpha\vert\alpha = f(\alpha)\rangle)$	unwinding
15	$\langle\alpha\vert\alpha = f(\alpha)\rangle =_{\mathsf{P}} f(\langle\alpha\vert\alpha = f(\alpha)\rangle)$	transitivity, 13, 14
16	$f(\langle\alpha\vert\alpha = f(\alpha)\rangle) =_{\mathsf{P}} \langle\alpha\vert\alpha = f(\alpha)\rangle$	symmetry, 15
17	$\langle f(\alpha)\vert\alpha = \langle\beta\vert\beta = f(\alpha)\rangle\rangle =_{\mathsf{P}} \langle\alpha\vert\alpha = f(\alpha)\rangle$	transitivity, 12, 16

We say that an equation $s = t$ is provable or derivable:

$$\vdash s =_{\mathsf{P}} t$$

if there is a list of equations $E_1, E_2, \dots E_n$, where $E_n \equiv s = t$, such that each equation E_i is either an instantiation of an axiom in the proof system or the result of an instantiation of a rule where the instantiations of the premises of the rule are present in the list prior to E_i itself. Such a list will be called a derivation from now on.

The system is not minimal. As an example the reflexivity axiom is derived from the other axioms and rules:

$$\begin{array}{lll} 1 & \langle t|\rangle = t & \text{A4} \\ 2 & t = \langle t|\rangle & \text{R1 , 1} \\ 3 & t = t & \text{R2, 2 , 1} \end{array}$$

Example 4. In table 2 the formal proof of

$$\vdash \langle f(\alpha)|\alpha = \langle\beta|\beta = f(\alpha)\rangle\rangle =_{\mathsf{P}} \langle\alpha|\alpha = f(\alpha)\rangle$$

is given. The proof doesn't use any particular strategy.

5 Soundness

This section is rather technical and uses some notions from rewriting theory: If $\to$ is a relation $\twoheadrightarrow$ is the transitive reflexive closure. A normal form is an element a such that there is no b with $a \to b$. An element x is terminating in a relation $\to$ (notation $SN^{\to}(x)$) if there is no sequence $x \equiv x_0 \to x_1 \to x_2 \to \dots$.

Theorem 9 (soundness). *If* $\vdash s =_{\mathsf{p}} t$ *then* $\models s =_{\mathsf{p}} t$.

For the proof we will make use of a characterization of bisimulation on the syntactic level. We will give a sufficient condition for a relation on the occurrences of two NTG's. Such a relation will be called an extensible relation or a syntax level bisimulation.

Definition 10. If R is a relation on occurrences of symbols in two NTG's s and t we define $x\ R^{\to_0}\ y$ iff $\exists x', y' : x \twoheadrightarrow_0 x' \wedge y \twoheadrightarrow_0 y' \wedge x'\ R\ y'$

A relation R on occurrences of symbols in two NTG's s and t is extensible if it satisfies the following conditions:

- $\mathrm{lm}(s)\ R^{\to_0}\ \mathrm{lm}(t)$
- If $x\ R\ y$, $x \twoheadrightarrow_0 x'$ and $y \twoheadrightarrow_0 y'$ then $x'\ R^{\to_0}\ y'$
- If $x\ R\ y$ then $SN^{\to_0}(x)$ iff $SN^{\to_0}(y)$
- If $x\ R\ y$, x and y $\to_0$ normal forms then x and y are occurrences of the same symbol.
- If $x\ R\ y$, $x \to_i x'$ and $y \to_i y'$ $(i > 0)$ then $x'\ R^{\to_0}\ y'$.

Lemma 11. *If* R *is an extensible relation between* s *and* t *then* $s \;\underline{\leftrightarrow}\; t$.

Proof. For any relation R we define $C_1\ R^*\ C_2$ if $\exists x, y : x\ R\ y$, $\overline{x} = C_1$ and $\overline{y} = C_2$. If R is extensible we have that R^* is a bisimulation. Note that $R^* = \overline{(R^{\to_0})^*}$. This means that $root(s)\ R\ root(t)$, because $root(s) = \overline{\mathrm{lm}(s)}$, $root(t) = \overline{\mathrm{lm}(t)}$ and $\mathrm{lm}(s)\ R^{\to_0}\ \mathrm{lm}(t)$. If $C_1\ R^*\ C_2$ then $C_1 = \overline{x}$ and $C_2 = \overline{y}$ for x and y with $x\ R\ y$. If x and y have no $\to_0$ normal forms then the label of C_1 is $\bullet$ just as the label of C_2. If x and y have normal forms x' and y' respectively then $x'\ R^{\to_0}\ y'$ which means that $x'\ R\ y'$. This last statement means that x' and y' must be occurrences of the same symbol. This means that they are either free occurrences of a variable or occurrence of the same function symbol. If $C_1 \to_i D_1$ and $C_2 \to_i D_2$ $(i > 0)$ then there are $x'' \in D_1$ and $y'' \in D_2$ such that $x' \to_i x''$ and $y' \to_i y''$. This means that $x''\ R^{\to_0}\ y''$ so $\overline{x''}\ R^*\ \overline{y''}$ or $D_1\ R^*\ D_2$ as required.

With the syntactic characterization in hand we will now prove soundness.

Proof of Thm. 9. We will do this by induction on the length of the derivation. Suppose the last step was:

A1 The identity relation is a bisimulation.

A2 The underlying graphs of left- and right-hand sides are exactly the same.

A3 Substitution has three cases. The first one is

$$\langle C[\alpha] | \alpha = s, E\rangle =_{\mathsf{p}} \langle C[s] | \alpha = s, E\rangle \ .$$

We define R on occurrences as follows: $x\ R\ y$ if x and y are matching occurrences outside of the hole or matching occurrences of the s sub terms. This R satisfies the conditions of the lemma. See Ex. 5 for an instance of this construction. The second case is

$$\langle t | \gamma = C[\alpha], \alpha = s, E\rangle =_{\mathsf{p}} \langle t | \gamma = C[s], \alpha = s, E\rangle \ .$$

This case is almost identical to the previous one. The third case is

$$\langle t|\alpha = C[\alpha]E\rangle =_{\mathsf{p}} \langle t|\alpha = C[C[\alpha]], E\rangle \ .$$

For this last case we have to modify R slightly: We still have $x\ R\ y$ for matching occurrences in t and E, but we add matching occurrences in the $C[.]$ part and the right hand side α occurrences.

A4 It is trivial that $\langle s|\rangle \underline{\leftrightarrow}\ s$.

A5 The natural syntactic relation on occurrences works.

A6 The symbol occurrences of s^E form a natural subset of those of $s^{E,F}$ this gives a natural 'identity' relation R on the occurrences of s^E and $s^{E,F}$. When lifted this relation is a bisimulation.

R1 The definition of bisimilarity is symmetric.

R2 Given bisimulations R_1 for $s \underline{\leftrightarrow}\ t$ and R_2 for $t \underline{\leftrightarrow}\ u$ the composition $R = R_1 \circ R_2$ is a bisimulation:

- We have $\overline{\mathrm{lm}(s)}\ R_1\ \overline{\mathrm{lm}(t)}$ and $\overline{\mathrm{lm}(t)}\ R_2\ \overline{\mathrm{lm}(u)}$ so $\overline{\mathrm{lm}(s)}\ R\ \overline{\mathrm{lm}(u)}$.
- If $C_1\ R\ C_3$ then there is a C_2 such that $C_1\ R_1\ C_2$ and $C_2\ R_2\ C_3$. Because R_1 and R_2 are bisimulations we know that $\mathrm{label}(C_1) = \mathrm{label}(C_2)$ and $\mathrm{label}(C_2) = \mathrm{label}(C_3)$ and so $\mathrm{label}(C_1) = \mathrm{label}(C_3)$.
- If $\overline{x}\ R\ \overline{y}$, $x \rightarrow_i x'$ and $y \rightarrow_i y'$ then there is a C such that $\overline{x}\ R_1\ C$ and $C\ R_2\ \overline{y}$. because the labels of $\overline{x}$ and C are the same C can be written as $C = \overline{z}$ with $z \rightarrow_i z'$. We then know $\overline{x'}\ R_1\ \overline{z'}$ and $\overline{z'}\ R_2\ \overline{y'}$ so $\overline{x'}\ R\ \overline{y'}$

R3 Given a bisimulation R between s and t and a context $C[.]$ we can define S on occurrences as $x\ S\ x$ if x is an occurrence in $C[.]$ and xSy if x is an occurrence in s and y one is t such that $\overline{x}\ R\ \overline{y}$. The relation S is extensible so $C[s] \underline{\leftrightarrow}\ C[t]$.

R4 We are given that

$$\models s_1 =_{\mathsf{p}} t_1\{s_1/\alpha_1, \ldots, s_n/\alpha_n\}, \ldots, s_n =_{\mathsf{p}} t_n\{s_1/\alpha_1, \ldots, s_n/\alpha_n\} \ .$$

Let $R_1, \ldots, R_n$ be bisimulations that witness these equalities. We must now construct a bisimulation between s_1 and $\langle \alpha_1|\alpha_1 = t_1, \ldots, \alpha_n = t_n\rangle$.

On the syntactic level we can do this by defining a relation R on the symbol occurrences of s_1, …, s_n, $t_1\{s_1/\alpha_1, \ldots, s_n/\alpha_n\}$, …, $t_n\{s_1/\alpha_1, \ldots, s_n/\alpha_n\}$ and $\langle \alpha_1|\alpha_1 = t_1, \ldots, \alpha_n = t_n\rangle$: for any two symbols occurrences x in s_i and y in $t_i\{s_1/\alpha_1, \ldots, s_n/\alpha_n\}$ such that $\overline{x}\ R_i\ \overline{y}$ we have $x\ R\ y$. Because of maximal α-conversion the NTG $t_i\{s_1/\alpha_1, \ldots, s_n/\alpha_n\}$ is formed by replacing the free occurrences of α_j in t_i by s_j. For any occurrence x in $t_i\{s_1/\alpha_1, \ldots, s_n/\alpha_n\}$ that is not a replacement we have $x\ R\ y$, where y is the same occurrence in the t_i sub-NTG in $\langle \alpha_1|\alpha_1 = t_1, \ldots, \alpha_n = t_n\rangle$. For any replacement occurrence x we have $x\ R\ y$, where y is the matching occurrence in the s_j (assuming that x is in a replacement of α_j. See Ex. 5 for an example of this construction.

If we take the transitive closure of this R then restrict it to occurrences in s_1 and $\langle \alpha_1|\alpha_1 = t_1, \ldots, \alpha_n = t_n\rangle$ and then lift it to the level of nodes, the relation is a bisimulation. To prove this we will use the lemma again. By the fact that R_1 is a bisimulation:

$$\mathrm{lm}(s_1)\ R\ \mathrm{lm}(t_1\{s_1/\alpha_1, \ldots, s_n/\alpha_n\}) \ .$$

Because t_1 is guarded:

$$\mathrm{lm}(t_1\{s_1/\alpha_1, \ldots, s_n/\alpha_n)\ R\ \mathrm{lm}(t_1 \subset \langle \alpha_1 | \alpha_1 = t_1, \ldots, \alpha_n = t_n \rangle) \ .$$

By definition of $\rightarrow_0$

$$\mathrm{lm}(\langle \alpha_1 | \alpha_1 = t_1, \ldots, \alpha_n = t_n \rangle) \rightarrow_0^2 \mathrm{lm}(t_1 \subset \langle \alpha_1 | \alpha_1 = t_1, \ldots, \alpha_n = t_n \rangle) \ .$$

$\square$

Example 5. In Fig. 4 the auxiliary relation described in the proof of soundness is shown for $\langle \alpha | \alpha = f(\beta), \beta = f(\alpha) \rangle$ and $\langle f(\beta) | \alpha = f(\beta), \beta = f(\alpha) \rangle$.

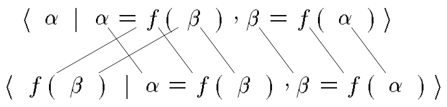

Fig. 4. soundness of substitution

In Fig. 5 the same is done for the constructed relation for the case of soundness. This picture looks rather complicated because of the many lines between

$$\langle \alpha | \alpha = f(f(\alpha)) \rangle \text{ and } f(\langle f(\alpha) | \alpha = f(f(\alpha)) \rangle)$$

and between

$$\langle f(\alpha) | \alpha = f(f(\alpha)) \rangle \text{ and } f(\langle \alpha | \alpha = f(f(\alpha)) \rangle) \ .$$

This jungle of lines comes from the fact that between these pairs we use a syntax level bisimulation generated by a graph level bisimulation. The three line types (solid, dotted and dashed) are used to distinguish between lines starting in the top NTG from symbols belonging to different nodes in the underlying graph.

6 Completeness

To prove completeness we first show that any NTG may be proven equal to a NTG of a special form: a flat NTG. We then prove completeness for term graphs only and conclude that because of soundness we also have proven completeness for arbitrary NTG's. The proof uses a derived axiom:

$$\langle s | \alpha = \beta, E \rangle =_{\mathsf{p}} \langle s\{\beta/\alpha\} | E\{\beta/\alpha\} \rangle \text{ if } \alpha \not\equiv \beta \ .$$

The idea is that we perform a simultaneous substitution of all occurrences of α by β and then apply garbage collection. This is the same as a sequence of substitution steps followed by garbage collection combined by using transitivity. The formal proof is an induction on the number of occurrences of α and is left to the reader.

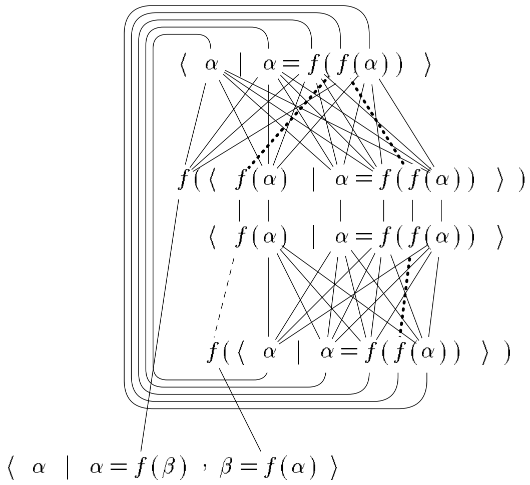

Fig. 5. soundness of folding

Theorem 12 (completeness). *If* $\models s =_{\mathsf{p}} t$ *then* $\vdash s =_{\mathsf{p}} t$.

Lemma 13. *Every NTG can be proven equal to a flat NTG.*

Proof. We will prove this statement by structural induction.

- If the given NTG is a variable α then $\alpha =_{\mathsf{p}} \langle \alpha | \rangle$.
- If the given NTG is of the form $f(t_1, \ldots, t_n)$ we know that

  $$f(t_1, \ldots, t_n) =_{\mathsf{p}} f(\alpha_1^{E_1}, \ldots, \alpha_n^{E_n}) \ .$$

  Define $E = \cup_{i=1}^{n} E_i$ and we have that

  $$\langle \alpha | \alpha = f(\alpha_1, \ldots, \alpha_n), E \rangle =_{\mathsf{p}}$$

  $$=_{\mathsf{p}} f(\langle \alpha_1 | \alpha = f(\alpha_1, \ldots, \alpha_n), E \rangle, \ldots, \langle \alpha_n | \alpha = f(\alpha_1, \ldots, \alpha_n), E \rangle) =_{\mathsf{p}}$$

  $$=_{\mathsf{p}} f(\alpha_1^{E_1}, \ldots, \alpha_n^{E_n}) \ .$$

- Given a box construct $\langle s | \alpha_1 = t_1, \ldots, \alpha_n = t_n \rangle$ we know by induction that this NTG is provably equal to one of the form $\langle \beta_0^{F_0} | \alpha_1 = \beta_1^{F_1}, \ldots, \alpha_n = \beta_n^{F_n} \rangle$. This is provably equal to

  $$\langle \alpha_0 | \alpha_0 = \beta_0^{F_0}, \alpha_1 = \beta_1^{F_1}, \ldots, \alpha_n = \beta_n^{F_n} \rangle \equiv \alpha_0^{E} \ .$$

  In this NTG we could have for some β_i that $\beta_i \notin \mathrm{def}(F_i)$. If this is the case we transform $\alpha_i = \beta_i^{F_i}$ into $\alpha_i = \beta_i$ and use either the $\alpha = \beta$ rule or the $\alpha = \alpha$ rule followed by $\bullet = \langle \beta_i | \beta_i = \bullet \rangle$. This process is terminating so we may assume that for each β_i, $\beta_i \in \mathrm{def}(F_i)$.

For each variable $\beta_{ij} \in \mathrm{def}(F_i)$ we define $s_{ij} \equiv \beta_{ij}^{F_i E}$. For this s_{ij} we can prove $(\beta_{ij} = f(\gamma_1, \ldots, \gamma_m) \in F_i)$:

$$s_{ij} \equiv \beta_{ij}^{F_i E} =_{\mathsf{p}} f(\gamma_1^{F_i E}, \ldots, \gamma_m^{F_i E}) \ .$$

If $\gamma_i \in \mathrm{def}(F_i)$ then for some j_i we have $\gamma_i \equiv \beta_{ij_i}$ so $\gamma_1^{F_i E} \equiv s_{ij_i}$. If $\gamma_i \notin \mathrm{def}(F_i)$ and $\gamma_i \in \mathrm{def}(E)$ then $\gamma_i \equiv \alpha_{i_i}$ and $\gamma_1^{F_i E} = \beta_{i_i 0}^{F_{i_i} E} \equiv s_{i_i 0}$. If $\gamma_i \notin \mathrm{def}(F_i)$ and $\gamma_i \notin \mathrm{def}(E)$ then we have that γ_i is a free variable. This gives rise to an equation

$$s_{ij} = f(t_1, \ldots, t_m)$$

where each t_i is either a free variable or another $s_{i'j'}$. Folding over the set of s_{ij}'s yields the conclusion that the NTG we started with is equal to a flat NTG.

□

Example 6. We will give a detailed derivation showing that

$$\langle f(\langle\alpha|\alpha = f(\alpha, \beta), \beta = \gamma\rangle, \delta)|\delta = \delta\rangle$$

is equal to some flat NTG. In that derivation the strategy of the proof is used. The level of indentation indicates the depth of recursion and the NTG between brackets on each level of indentation is the NTG we are proving equal to a term graph at that point.

$[\langle f(\langle\alpha|\alpha = f(\alpha, \beta), \beta = \gamma\rangle, \delta)|\delta = \delta\rangle]$
  $[f(\langle\alpha|\alpha = f(\alpha, \beta), \beta = \gamma\rangle, \delta)]$
    $[\langle\alpha|\alpha = f(\alpha, \beta), \beta = \gamma\rangle]$
      $[\alpha]\ \alpha = \langle\alpha|\rangle$
      $[f(\alpha, \beta)]$
        $[\alpha]\ \alpha = \langle\alpha|\rangle$
        $[\beta]\ \beta = \langle\beta|\rangle$
        $f(\alpha, \beta) = f(\langle\alpha|\rangle, \langle\beta|\rangle) = \langle\gamma|\gamma = f(\alpha, \beta)\rangle$
      $[\gamma]\ \gamma = \langle\gamma|\rangle$
      $\langle\alpha|\alpha = f(\alpha, \beta), \beta = \gamma\rangle =$
        $= \langle\delta|\delta = \langle\alpha|\rangle, \alpha = \langle\gamma'|\gamma' = f(\alpha, \beta)\rangle, \beta = \langle\gamma|\rangle\rangle =$
        $= \langle\delta|\delta = \alpha, \alpha = \langle\gamma'|\gamma' = f(\alpha, \beta)\rangle\beta = \langle\gamma|\rangle\rangle =$
        $= \langle\alpha|\alpha = \langle\gamma'|\gamma' = f(\alpha, \beta)\rangle\beta = \langle\gamma|\rangle\rangle =$
        $= \langle\alpha|\alpha = \langle\gamma'|\gamma' = f(\alpha, \gamma)\rangle\rangle$
      Giving the equation:
      $\langle\langle\gamma'|\gamma' = f(\alpha, \gamma)\rangle|\alpha = \langle\gamma'|\gamma' = f(\alpha, \gamma)\rangle\rangle =$
        $= f(\langle\langle\gamma'|\gamma' = f(\alpha, \gamma)\rangle|\alpha = \langle\gamma'|\gamma' = f(\alpha, \gamma)\rangle\rangle, \gamma)$
      So after folding
      $\langle\alpha|\alpha = f(\alpha, \beta), \beta = \gamma\rangle = \langle\alpha|\alpha = f(\alpha, \gamma)\rangle$
    $[\delta]\ \delta = \langle\delta|\rangle$

$$f(\langle\alpha|\alpha = f(\alpha,\beta), \beta = \gamma\rangle, \delta) = f(\langle\alpha|\alpha = f(\alpha,\gamma)\rangle, \langle\delta|\rangle) =$$
$$= \langle\beta|\beta = f(\alpha,\delta), \alpha = f(\alpha,\gamma)\rangle$$
$$[\delta]\ \delta = \langle\delta|\rangle$$
$$\langle f(\langle\alpha|\alpha = f(\alpha,\beta), \beta = \gamma\rangle, \delta)|\delta = \delta\rangle =$$
$$= \langle\epsilon|\epsilon = \langle\beta|\beta = f(\alpha,\delta), \alpha = f(\alpha,\gamma)\rangle, \delta = \langle\delta|\rangle\rangle =$$
$$= \langle\epsilon|\overbrace{\epsilon = \langle\beta|\underbrace{\beta = f(\alpha,\delta), \alpha = f(\alpha,\gamma)}_{E_1}\rangle, \delta = \langle\delta'|\underbrace{\delta' = \bullet}_{E_2}\rangle}^{E}\rangle$$

Leading to the recursion equations

$$\langle\beta^{E_1}|E\rangle = f(\langle\alpha^{E_1}|E\rangle, \langle\delta'^{E_2}|E\rangle)$$
$$\langle\alpha^{E_1}|E\rangle = f(\langle\alpha^{E_1}|E\rangle, \gamma)$$
$$\langle\delta'^{E_2}|E\rangle = \bullet$$

Folding yields:

$$\langle f(\langle\alpha|\alpha = f(\alpha,\beta), \beta = \gamma\rangle, \delta)|\delta = \delta\rangle = \langle\alpha|\alpha = f(\beta,\delta), \beta = f(\beta,\gamma), \delta = \bullet\rangle$$

Proof of Thm. 12. Given $\models s = t$ we know that there are flat NTG's s' and t' such that $\vdash s = s'$ and $\vdash t = t'$, so $\models s' = t'$. If we can show $\vdash s' = t'$, it follows that $\vdash s = t$.

A flat NTG has the special property that there is a one to one correspondence between the nodes of the graph and the variables occurring. The given bisimulation R between s' and t' can therefore be seen as a relation on the variables occurring. Using that observation for every pair $(\alpha,\beta) \in R$ we define a variable $\gamma_{(\alpha,\beta)}$. If α is a free variable then $\gamma_{(\alpha,\beta)} \equiv \alpha$. If α is bound and $\alpha = f(\alpha_1,\ldots,\alpha_n)$ and $\beta = f(\beta_1,\ldots,\beta_n)$ are the definitions of α and β then we also define a term

$$t_{(\alpha,\beta)} \equiv f(\gamma_{(\alpha_1,\beta_1)},\ldots,\gamma_{(\alpha_n,\beta_n)})\ .$$

Due to the fact that R is a bisimulation, the variables in $t_{(\alpha,\beta)}$ are all defined. Assume $s' \equiv \alpha_0^E$ and $t' \equiv \beta_0^F$. We define

$$r = \langle\gamma_{(\alpha_0,\beta_0)}|\gamma_{(\alpha,\beta)} = t_{(\alpha,\beta)} \text{ for } (\alpha,\beta) \in R \text{ with } t_{(\alpha,\beta)} \text{ defined}\rangle$$

For all α defined in s', we can derive:

$$\alpha^E = f(\alpha_1^E,\ldots,\alpha_n^E)$$

which, defining $u_{(\alpha,\beta)} \equiv \alpha^E$, can be read as:

$$u_{(\alpha,\beta)} = f(u_{(\alpha_1,\beta_1)},\ldots,u_{(\alpha_n,\beta_n)}) = t_{(\alpha,\beta)}\{u_{(\alpha_i,\beta_i)}/\gamma_{(\alpha_i,\beta_i)}\}_{i=1}^n$$

for each β such that $\alpha R \beta$. This β is defined as $\beta = f(\beta_1,\ldots,\beta_n)$. So by folding:

$$\vdash s' = r\ .$$

In the same way one can derive $r = t'$ and so

$$\vdash s' = t'\ .$$

□

Example 7. We will now prove the equality of two flat NTG's using the strategy of the previous proof. We have that

$$\models s \equiv \alpha_0^{\{\alpha_0=f(\alpha_0,\beta,\alpha_1),\alpha_1=f(\alpha_1,\beta,\alpha_0)\}} = \alpha_0^{\{\alpha_0=f(\alpha_1,\beta,\alpha_0),\alpha_1=f(\alpha_0,\beta,\alpha_1)\}} \equiv t$$

via the bisimulation $R = \{(\alpha_0, \alpha_0), (\alpha_0, \alpha_1), (\alpha_1, \alpha_0), (\alpha_1, \alpha_1), (\beta, \beta)\}$.

We define

$$r \equiv \alpha_{00}^{\{\alpha_{00}=f(\alpha_{01},\beta,\alpha_{10}),\alpha_{01}=f(\alpha_{00},\beta,\alpha_{11}),\alpha_{10}=f(\alpha_{11},\beta,\alpha_{00}),\alpha_{11}=f(\alpha_{10},\beta,\alpha_{01})\}} \quad .$$

Putting $E = \{\alpha_0 = f(\alpha_0, \beta, \alpha_1), \alpha_1 = f(\alpha_1, \beta, \alpha_0)\}$ we can derive

$$\alpha_0^E = f(\alpha_0^E, \beta, \alpha_1^E)$$

and

$$\alpha_1^E = f(\alpha_1^E, \beta, \alpha_0^E) \quad .$$

If we define

$$s_{ij} = \alpha_i^E \text{ for } i, j = 0, 1$$

we can read the derived equations as

$$\begin{aligned} s_{00} &= f(s_{01}, \beta, s_{10}) \\ s_{01} &= f(s_{00}, \beta, s_{11}) \\ s_{10} &= f(s_{11}, \beta, s_{00}) \\ s_{11} &= f(s_{10}, \beta, s_{01}) \end{aligned}$$

and fold then into

$$s_{00} = \alpha_{00}^{\{\alpha_{00}=f(\alpha_{01},\beta,\alpha_{10}),\alpha_{01}=f(\alpha_{00},\beta,\alpha_{11}),\alpha_{10}=f(\alpha_{11},\beta,\alpha_{00}),\alpha_{11}=f(\alpha_{10},\beta,\alpha_{01})\}} \quad .$$

Thus we have derived $s = r$. In a similar way $t = r$ can be derived. Hence $s = t$.

7 Acknowledgments

The author thanks Zena Ariola for making possible his stay at the University of Oregon, where part of the work on the final revision of this paper was done. The author thanks Zena Ariola and Jan Willem Klop for commenting on drafts of this paper.

References

1. Z. M. Ariola and J. W. Klop. Equational Term Graph Rewriting. *To appear in Fundamenta Informaticae. Extended version: CWI Report CS-R9552*, 1995.
2. Z.M. Ariola and J.W. Klop. Cyclic Lambda Graph Rewriting. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 416–425, Paris, France, July 1994. IEEE Computer Society Press.
3. H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* North-Holland, Amsterdam, 1984.
4. H.P. Barendregt, M.J.C.D. van Eekelen, J.R.W. Glauert, J.R. Kennaway, M.J. Plasmeijer, and M.R. Sleep. Term Graph Reduction. In *Proc. Conference on Parallel ARchitecture and Languages Europe (PARLE)*, pages 141–158, Eindhoven, The Netherlands, 1987. Springer-Verlag LNCS 259.
5. E. Barendsen. *Types and Computations in Lambda Calculi and Graph Rewrite Systems.* PhD thesis, University of Nijmegen, 1995.
6. S.L. Bloom and Z. Esik. *Iteration Theories.* EATCS Monograph. Springer, 1993.
7. C.C. Elgot. Monadic Computation and Iterative Algebraic Theories. In H. E. Rose and J. C. Shepherdson, editors, *Logic Colloquium '73 (Bristol, England)*, number 80 in Studies in Logic and the Foundations of Mathematics, pages 175–230. North Holland, 1975.
8. J. W. Klop. Term Rewriting Systems. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume II, pages 1–116. Oxford University Press, 1992.
9. R. Milner. A complete inference system for a class of regular behaviours. *JCSS*, 28:227–247, 1984.
10. S. L. Peyton Jones. *The implementation of Functional Programming Languages.* Prentice-Hall International, Englewood Cliffs, N.J., 1987.
11. M.R. Sleep, M.J. Plasmeijer, and M.C.D.J. van Eekelen, editors. *Term Graph Rewriting: Theory and Practice.* John Wiley & Sons, 1993.
12. S. Smetsers. *Graph Rewriting and Functional Languages.* PhD thesis, University of Nijmegen, 1993.
13. G. Ştefănescu. The Algebra of Flownomials Part 1: Binary Flownomials; Basic Theory. Technical report, Technical University Munich, November 1994.