Proof Systems for Nested Term Graphs

S.C.C. Blom

Department of Mathematics and Computer Science,
Vrije Universiteit, de Boelelaan 1081, 1081 HV Amsterdam.

This report was written in partial fulfilment of the requirements for my Master's degree, under the supervision of prof.dr J.W. Klop.

March 10, 1994.

Abstract

In this report we present some proof systems for possibly cyclic term graphs over a first-order signature. This is done first for 'flat' term graphs (which are just systems of recursion equations) and next for 'nested' term graphs. In the latter, the recursion construct may be nested. We prove soundness and completeness of the proof systems. The completeness refers to equivalence with respect to tree unwinding, which is the same as bisimulation equivalence. Bisimulation equivalence is generated by the copying transformation. In the first part of the paper we consider sequential copying, and show that this notion is strictly weaker than general copying.

Note

This report was written in partial fulfilment of the requirements for the Master's degree, under the supervision of prof.dr J.W. Klop.

Contents

Introduction

In implementing functional programming languages the issue of term graph rewriting is of great importance. Using graphs instead of terms we are able to share subterms, thus avoiding repeated computations of the same subterm. Moreover, by means of cyclic graphs we are able to deal with infinite objects (the tree unwindings) in a finitary way.

As the underlying semantics of (possibly cyclic) term graphs over a first-order signature we take the set of finite and infinite trees over that signature. Term graphs are equivalent (have the same semantics) if they unshare or unwind to the same tree. This equivalence is generated by the copying transformation, which is the most fundamental operation on term graphs. This operation is studied in [2]. It is in fact the same as an inverse rooted homomorphism between graphs; see [8]. In [2] it is pointed out that the equivalence of having the same tree unwinding is also conveniently described by the notion of bisimulation, which is well-known from the theory of concurrent and communicating processes. The (inverse of) the copying transformation is then the same as a so-called functional bisimulation. Copying can be syntactically described by introducing some new node names, and then deriving by ordinary equational logic manipulations a new set of recursion equations, i.e. a new graph. In [2] the question is raised whether a copying step can be analyzed in a sequence of simpler steps, namely sequential copying steps in which only one new node is introduced in each step. We solve this question in our first chapter by showing that it cannot: general copying (in which nodes are created in parallel) is stronger than (the transitive closure of) sequential copying. For acyclic term graphs though, sequential copying is just as strong as general copying. We leave open the question whether sequential copying is confluent, like general copying is. This seems to be a difficult puzzle.

In the second chapter we turn to the main subject of this report: developing a proof theory for term graphs. It should be noted that nowhere in this work we are concerned with rewriting of term graphs; we are just concerned with axiomatizing the equivalence of term graphs explained before. This paper reports some initial steps, and by no means we present a complete survey of the situation. Our starting point is the observation that a term graph, or a system of recursion equations, is a particularly simple case of a Recursive Program Scheme ([4], [5], [3]), namely one where the recursion variables are in fact constants in the representation as a RPS. Now for RPSs Courcelle and Vuillemin have given a complete proof system in [4]. We specialize their system to the present simple case. In doing so we found that the proof system by Courcelle and Vuillemin is not correct; the mistake went undetected for seventeen years, as we checked with the authors. However, it seems easily to be repairable.

Next, we turn our attention to nested term graphs. The relevance of nesting for term graphs is demonstrated in work of Ariola and Klop([2]): for unnested (flat) term graphs non-confluence phenomena crop up when notions of copying and rewriting are applied too liberally. Nesting of term graphs imposes a hierarchical structure on graphs by means of which it is possible to have restrained ways of copying and rewriting that are confluent. Especially this part of our work enters unknown terrain: for nested term graphs, there is not yet a proof theory. We present also for nested term graphs a proof system that is sound and complete; however, our treatment here is still tentative as it reports work still in progress. The proof of soundness and completeness is sketched in detail but is as yet to be elaborated. Also the design of the last proof system is still subject to experimentation, in order to arrive at a 'minimal' system or to find the most elegant presentation. This is a goal we hope to realize in future work.

Chapter 1

Terms, term graphs and trees.

In this chapter three extensions of the well-known concept of term are given: term graphs, nested term graphs and trees. As one would expect the set of trees is a subset of the set of term graphs, which is a subset of the set of nested term graphs. With suitable equivalence relations for term graphs and nested term graphs, the sets modulo these equivalences become isomorphic to the set of trees. These equivalence relations can be characterized by bisimulation. The notion of bisimulation gives a means of handling the graphs; it can be characterized directly, as in [2], or indirectly, using the notion of copying, which is also a relation on graphs. A simplified version of copying is sequential copying. Sequential copying however is substantially weaker than copying, making it impossible to use it as a tool in proofs, except in simple cases that do not occur in this thesis. Some of the work on sequential copying cannot be understood without knowledge of [2].

1.1 Syntax.

1.1.1 Term graphs.

By convention $\mathcal{V}$ will be a set of variables $(\alpha, \beta, \ldots)$ and $\mathcal{F}$ will be a set of function symbols $(f, F, g, G, \ldots)$ each with a given arity. The set of functions includes two special constants: $\bullet$ and $\perp$. They will be treated later on.

Definition 1.1.1

- A *constant* is a function symbol with arity 0.
- A *term* is recursively defined as:
  1. α, for $\alpha \in \mathcal{V}$.
  2. $f(t_1, \ldots, t_n)$, for $f \in \mathcal{F}$ with arity n and t_i a term $(i = 1, \ldots, n)$.
- A *guarded* term is a term that is not just a variable.
- A *context* $C[.]$ is a term with a single occurrence of $\Box$, the symbol denoting an open place.
- $C[t]$ stands for the context $C[.]$ with $\Box$ replaced by the term t.

- A *program* $\mathcal{P}$ is a set of equations, whose left-hand sides are pairwise different variables and whose right-hand sides are guarded terms.
- A variable occurring as the left-hand side of some equation in $\mathcal{P}$ is called *defined* in $\mathcal{P}$. The equation is referred to as the *defining equation* of the variable and the right-hand side as its definition.
- If α is defined in $\mathcal{P}$ then $\mathcal{P}@\alpha$ is the smallest program $\mathcal{P}' \subset \mathcal{P}$ such that α is defined in $\mathcal{P}'$ and if a certain β is defined in $\mathcal{P}'$, γ occurs in the definition of β, and γ is defined in $\mathcal{P}$ then γ is defined in $\mathcal{P}'$.
- A *term graph* is a program $\mathcal{P}$ with a root α such that $\mathcal{P}@\alpha = \mathcal{P}$. The root can be explicitly mentioned as in:

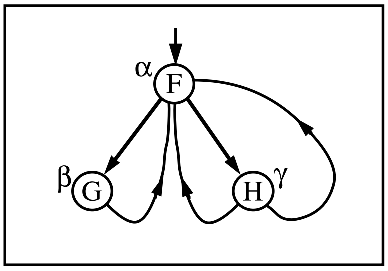
  $$\{\alpha | \alpha = f(\beta), \beta = g(\alpha, \beta)\} \text{ or } \{\alpha = f(\beta), \beta = g(\alpha, \beta)\}@\alpha$$

  or can be implicit (as the first variable):

  $$\{\alpha = f(\beta), \beta = g(\alpha, \beta)\} \equiv \{\alpha | \alpha = f(\beta), \beta = g(\alpha, \beta)\}$$

- A variable is *fresh* for $\mathcal{P}$ if it doesn't occur, it is *free* if it occurs, but is not defined.

Example 1.1.1 A term graph can be conveniently pictured as a rooted, labeled directed graph, as in the figure below.



$$\{\alpha = F(\beta, \gamma), \beta = G(\alpha), \gamma = H(\alpha, \alpha)\}$$

1.1.2 Nested term graphs.

The special constant $\bullet$ will be called *black hole.*

Definition 1.1.2

- A *nested term graph*, or *NTG*, is recursively defined as:
  1. t, for any term t as defined in the previous section.
  2. $\langle s | \alpha_1 = t_1, \ldots, \alpha_n = t_n \rangle$, for NTG's s and t_i $(i = 1, \ldots, n)$ and pairwise different variables α_i $(i = 1, \ldots, n)$.
- An *environment* E is the second part of a nesting construct. That is: the E in $\langle t | E \rangle$ is the environment. Formally an environment is a set of equations with pairwise different variables as the left-hand sides. Programs therefore are special environments.

- If $E = \{\alpha_1 = t_1, \ldots, \alpha_n = t_n\}$ then we use the notations:

$$\langle s|\alpha_1 = t_1, \ldots, \alpha_n = t_n\rangle \equiv \langle s|E\rangle \equiv s^E$$

  In view of this notation we will sometimes refer to an environment as *exponent*.
  $E = \{\alpha_1 = t_1, \ldots, \alpha_n = t_n\}$ will also be written as $E = \{\vec{\alpha} = \vec{t}\}$.
  In the same way:

$$s(t_1, \ldots, t_n) \equiv s(\vec{t})$$
$$s_1(\vec{t}), \ldots, s_n(\vec{t}) \equiv \vec{s}(\vec{t})$$

- A NTG is *guarded* if it is neither a variable (α) nor a nesting construct ($\langle t|E\rangle$).

- A variable is fresh for an object if it doesn't occur in that object, where the object can be a NTG or an environment.

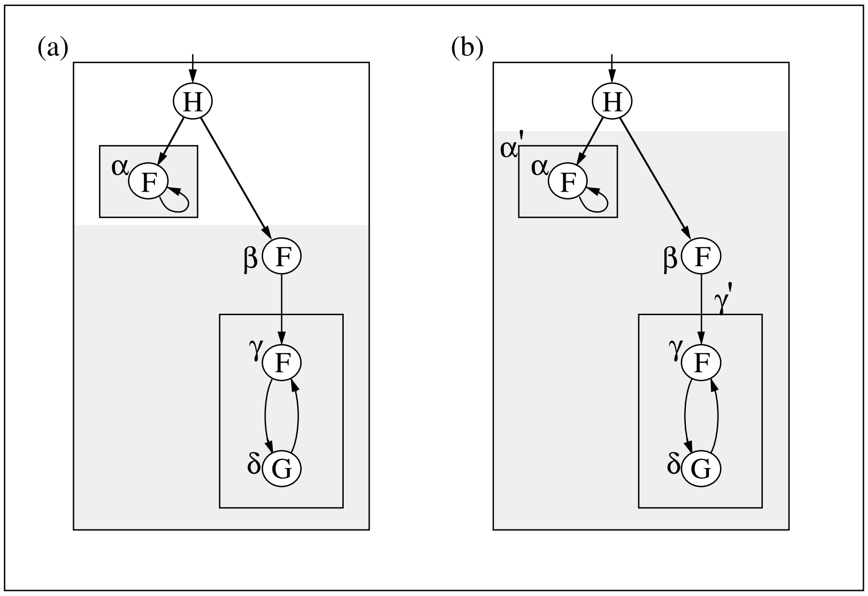
- A *context* $C[.]$ is a NTG in which a free variable that occurs only once has been replaced by the symbol $\square$. So $\langle\square|\alpha = f(\alpha)\rangle$ and $\langle\alpha|\alpha = \square\rangle$ are contexts, but $\langle\square|\alpha = \square\rangle$ and $\langle\alpha|\square = f(\alpha)\rangle$ are not.

- By $C[t]$ we mean $C[.]$ with $\square$ replaced by a NTG t.

- α*-conversion* ($=_\alpha$). If α is defined in E, and β is fresh for s^E then $s^E =_\alpha t^F$, where t and F are the result of replacing every occurrence of α by β in s and E. Also:

  1. $s =_\alpha t \Rightarrow C[s] =_\alpha C[t]$
  2. $r =_\alpha s, s =_\alpha t \Rightarrow r =_\alpha t$
  3. $s =_\alpha t \Rightarrow t =_\alpha s$

Example 1.1.2 In the following picture two NTG's are pictured. The boxes represent nesting constructs, the environment parts of those constructs are shaded.



(a) $\langle H(\langle\alpha|\alpha = F\alpha\rangle, \beta)|\beta = F(\langle\gamma|\gamma = F(\delta), \delta = G(\gamma)\rangle)\rangle$
(b) $\langle H(\alpha', \beta)|\alpha' = \langle\alpha|\alpha = F\alpha\rangle, \beta = F(\gamma'), \gamma' = \langle\gamma|\gamma = F(\delta), \delta = G(\gamma)\rangle\rangle$

Definition 1.1.3 A *substitution* σ is a function from a subset of $\mathcal{V}$ to the set of NTG's, such that if $\alpha \in D_\sigma$ (the domain of σ) then $\sigma(\alpha) \neq \alpha$. It extends to general NTG's in the following way: If necessary apply α-conversion, to ensure that none of the bound variables in the NTG that is to be substituted, is identical to a variable occurring free in any $\sigma(\alpha)$.

1. $\alpha\sigma = \sigma(\alpha)$, if $\alpha \in D_\sigma$.
2. $\alpha\sigma = \alpha$, if $\alpha \notin D_\sigma$.
3. $f(\vec{t})\sigma = f(\vec{t}\sigma)$
4. $\langle t|\vec{\alpha} = \vec{s}\rangle\sigma = \langle t\sigma{\uparrow}_V \; |\vec{\alpha} = \vec{s}\sigma{\uparrow}_V\rangle$, where $V = \{\vec{\alpha}\}^c$.

By $S\{t/\alpha\}$ we mean the result of substituting t for α in S.

Example 1.1.3

$$\langle\alpha|\alpha = f(\alpha,\beta)\rangle\{\gamma/\beta,\gamma/\alpha\} = \langle\alpha\{\gamma/\beta\}|\alpha = f(\alpha,\beta)\{\gamma/\beta\}\rangle =$$

$$\langle\alpha|\alpha = f(\alpha\{\gamma/\beta\},\beta\{\gamma/\beta\})\rangle = \langle\alpha|\alpha = f(\alpha,\gamma)\rangle$$

1.1.3 Infinite Trees

The set of possibly infinite trees over the first-order signature that is considered, will provide us with the semantics of the syntactic constructs of term graph and NTG introduced above. For any set V the set V^* is the set of strings over V. Assume $\mathbb{N}$, $\mathcal{F}$ and $\mathcal{V}$ pairwise disjoint. A *tree* is a subset T of $(\mathbb{N} \cup \mathcal{F} \cup \mathcal{V})^*$ satisfying:

- Each $t \in T$ can be written as wx where $w \in \mathbb{N}^*$ and $x \in \mathcal{F} \cup \mathcal{V}$.
- $wnx \in T$ for some $x \in \mathcal{V} \cup \mathcal{F}$ iff $wf \in T$ and $1 \leq n \leq arity(f)$.
- if $wx, wy \in T$ then $x = y$.

Put $T_\perp = \{wx \in T | x \neq \perp\}$.
From $T_\perp$ one can reconstruct the original T, as follows:

$$T = T_\perp \cup \{wn \perp | wf \in T_\perp, 1 \leq n \leq arity(f), \forall x : wnx \notin T_\perp\}$$

Define:

$$S \leq T \Leftrightarrow S_\perp \subset T_\perp$$

A *substitution* σ is a function from $\mathcal{V}$ to the set of trees, extending to trees in the following way:

$$T\sigma = \{wx \in T | x \in f\} \cup \{wv | wx \in T; x \in \mathcal{V}; v \in \sigma(x)\}$$

Proposition 1.1.1 *For trees S and T:*

$$S = T \Leftrightarrow S \leq T \wedge T \leq S$$

Proof If $S = T$ then of course $S_\perp = T_\perp$ and so $S \leq T \wedge T \leq S$
If $S \leq T \wedge T \leq S$ then $S_\perp = T_\perp$ and so $S = T$.

□

Proposition 1.1.2 *For trees S, T and U:*

$$S \leq T \text{ and } T \leq U \Rightarrow S \leq U$$

Proof

$$S \leq T \wedge T \leq U \Leftrightarrow S_\perp \subset T_\perp \wedge T_\perp \subset U_\perp \Rightarrow S_\perp \subset U_\perp \Leftrightarrow S \leq U$$

□

Definition 1.1.4 Let $f \in \mathcal{F}$ and trees T_i $(i = 1, \ldots, n)$ be given. Let the arity of f be n. Define:

$$f(T_1, \ldots, T_n) = \{f\} \cup_{i=1}^{n} \{iw | w \in T_i\}$$

Also define $\alpha = \{\alpha\}$ for $\alpha \in \mathcal{V}$.

These definitions give an embedding of the set of terms in the set of trees and make it possible to say that a tree T is either of the form α or of the form $f(\vec{T})$, where $\vec{T}$ is a vector of trees with length matching the arity of f.
So $f(\alpha, g(\beta))$ stands for:

$$\{f, 1\alpha, 2g, 21\beta\}$$

Proposition 1.1.3 *For vectors of trees $\vec{S}$ and $\vec{T}$:*

$$\vec{S} \leq \vec{T} \Leftrightarrow f(\vec{S}) \leq f(\vec{T})$$

Proof Trivial.

□

Definition 1.1.5 A *context* $C[.]$ is a tree in which a single nullary symbol from $\mathcal{V}$ or $\mathcal{F}$ is replaced by the symbol $\square$. $C[T]$ stands for

$$(C[.] \setminus \{v\square\}) \cup \{vw | w \in T\}, \text{ where } v\square \in C[.]$$

On these trees we can define a notion of prefix and with that notion of prefix a notion of limit, such that if a sequence shares an ever increasing prefix, it has a limit.

Definition 1.1.6

- The *prefix* of length n of a tree T, denoted $T{\uparrow_n}$ is:

$$T{\uparrow_n} = \{w \in T || w| \leq n\}, \text{ where } |.| \text{ is the length of the string } w.$$

- We say that S is the *limit* of a sequence of trees S_n

$$\lim_{n\to\infty} S_n = S$$

if

$$\forall n : \exists N : \forall m > N : S_m{\uparrow_n} = S{\uparrow_n}$$

1.1.4 Term rewrite systems

An expression can be either a term, a nested term graph or a tree. Note that the set of terms is embedded in the set of expressions.

Definition 1.1.7

- A *rule* $s \to t$ has a left-hand side s and a right-hand side t. Both s and t must be terms, s must be guarded and t does not contain a variable that does not occur in s.
- A *term rewrite system* or *TRS* is a set of rules.
- A term s rewrites to a term t by a rule $r \equiv r_l \to r_r$, notation $s \to_r t$, if

  $$s \equiv C[r_l\sigma] \text{ and } t \equiv C[r_r\sigma]$$

  for some substitution σ and a context $C[.]$.
- A term s rewrites to a term t in a TRS $\mathcal{R}$, notation $s \to_{\mathcal{R}} t$, iff for some $r \in \mathcal{R}$: $s \to_r t$. If there is no confusion the notation $s \to t$ is often used.
- All the rewrite arrows are relations. The transitive reflexive closure of an arrow $\to$ is denoted by $\twoheadrightarrow$.

If the set of expressions is the set of terms, then the definition of TRS coincides with the traditional definitions. In the other cases it is a real extension.
For more information on term rewriting see [6], [7].

1.1.5 Connections

We mention some obvious connections between the concepts introduced thus far.
A term graph $\mathcal{P}@\alpha$ can be seen as a nested term graph $\alpha^{\mathcal{P}}$.
The set of finite trees and the set of terms are isomorphic.
Terms and therefore finite trees are also term graphs and nested term graphs.
A tree T can be seen as a term graph:

$$\{\alpha_w = x(\alpha_{w1}, \ldots, \alpha_{wn}) | wx \in T\}@\alpha_\epsilon$$

and also as a nested term graph:

$$\langle \alpha_\epsilon | \{\alpha_w = x(\alpha_{w1}, \ldots, \alpha_{wn}) | wx \in T\} \rangle$$

If T is infinite, the definitions of term graph and nested term graph have to be stretched to include infinite programs/environments.
So $f(\alpha, g(\beta))$ can be seen as the NTG:

$$\langle \alpha_\epsilon | \alpha_\epsilon = f(\alpha_1, \alpha_2), \alpha_1 = \alpha, \alpha_2 = g(\alpha_{21}), \alpha_{21} = \beta \rangle$$

but also as the NTG:

$$f(\alpha, g(\beta))$$

So in a proof system that deals with nested term graphs it should be possible to prove that the two are equal. This will indeed be the case for the proof systems that we introduce in the sequel.

1.2 Copying and bisimulation.

1.2.1 Definitions.

Definition 1.2.1

- Copying on programs. A *basic copying step* has three components:
  1. replace one or more variables α by $\alpha + \beta$ in which β is fresh.
  2. replace every equation $s + t = T$ by $s = T, t = T$.
  3. rewrite every right-hand side to a normal form of the TRS with rules $x + y \to x$ and $x + y \to y$.
- A basic step is *sequential*, if in part (1) only a single new variable is introduced.
- A program $\mathcal{P}$ *(sequentially) copies to* $\mathcal{Q}$ if $\mathcal{Q}$ is the result of repeated (sequential) basic steps. Notation: $\mathcal{P} \to_c \mathcal{Q}$ ($\mathcal{P} \to_{1c} \mathcal{Q}$).
- Copying on term graphs. A basic (sequential) step from $\mathcal{P}@\alpha$ to $\mathcal{Q}@\beta$ is a basic (sequential) step from $\mathcal{P}$ to $\mathcal{Q}$ satisfying $\mathcal{P}@\alpha = \mathcal{P}$, $\mathcal{Q}@\beta = \mathcal{Q}$ and $\alpha \equiv \beta$.
- A term graph $\mathcal{P}@\alpha$ (sequentially) copies to $\mathcal{Q}@\alpha$ if $\mathcal{Q}@\alpha$ is the result of repeated (sequential) basic steps. Notation: $\mathcal{P}@\alpha \to_c \mathcal{Q}@\alpha$ ($\mathcal{P}@\alpha \to_{1c} \mathcal{Q}@\alpha$).

Example:

$$\{\alpha | \alpha = F(\alpha, \alpha)\}$$

goes to

$$\{\alpha | (\alpha + \alpha') + \alpha'' = F((\alpha + \alpha') + \alpha'', (\alpha + \alpha') + \alpha'')\}$$

in the first step, goes to

$$\begin{array}{lll} \{\alpha & | & \alpha = F((\alpha + \alpha') + \alpha'', (\alpha + \alpha') + \alpha'') \\ & , & \alpha' = F((\alpha + \alpha') + \alpha'', (\alpha + \alpha') + \alpha'') \\ & , & \alpha'' = F((\alpha + \alpha') + \alpha'', (\alpha + \alpha') + \alpha'')\} \end{array}$$

in the second step and finally rewrites to:

$$\{\alpha | \alpha = F(\alpha', \alpha''), \alpha' = F(\alpha', \alpha), \alpha'' = F(\alpha, \alpha''\}$$

So:

$$\{\alpha | \alpha = F(\alpha, \alpha)\} \to_c \{\alpha | \alpha = F(\alpha'', \alpha''), \alpha' = F(\alpha', \alpha), \alpha'' = F(\alpha, \alpha''\}$$

This notion of copying coincides with that introduced by Z.M. Ariola and J.W. Klop in [2]. For both programs and term graphs: $\to_{1c} \subset \to_c$.

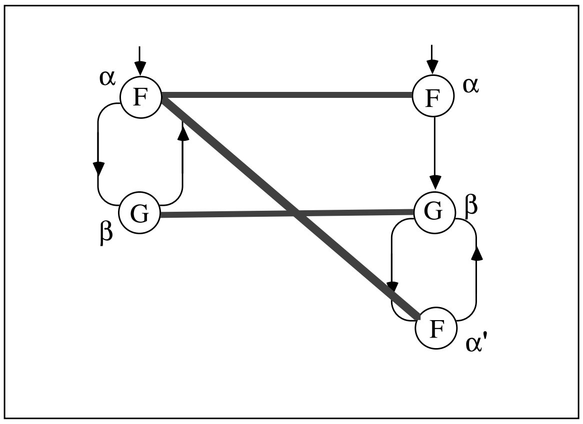
Definition 1.2.2

- A *bisimulation* is a relation R between the defined variables of two term graphs $\mathcal{P}@\alpha$ and $\mathcal{Q}@\beta$, satisfying:
  - $\alpha R \beta$ (the roots are related).

  – If $\alpha' R \beta'$ then $\alpha' = t(\vec{\alpha}) \in \mathcal{P}$, $\beta' = t(\vec{\beta})$, and $\vec{\alpha} R \vec{\beta}$.

- Two term graphs G and H are *bisimilar* ($G \underline{\leftrightarrow} H$) if there exists a bisimulation R from G to H.
- When replacing α by $\alpha + \beta$, α is said to be the *parent* of β and β is the *child* of α.
- α_0 and α_n are relatives if there are $\alpha_1, \ldots, \alpha_{n-1}$ such that for $i = 1, \ldots, n$ α_{i-1} is either child or parent of α_i.

Example 1.2.1 A bisimulation between two term graphs can be pictured drawing the two term graphs and connecting bisimilar nodes in the graph with a line.



$$\{\alpha|\alpha = F(\beta), \beta = G(\alpha)\} \underline{\leftrightarrow} \{\alpha|\alpha = F(\beta), \beta = G(\alpha'), \alpha' = F(\beta)\}$$

If $G \to_c H$ in a basic step, define the relation R between the defined variables of G and H as $\alpha R \beta$ if α is a relative of β. This R is a bisimulation, so the equivalence relations on term graphs induced by (sequential) copying are subsets of the equivalence relations induced by bisimilarity.

1.2.2 Sequential copying

The rest of this chapter answers some questions about sequential copying from [2]. These sections can be skipped without impairing the understanding of the sequel.

Sequential copying is simpler than copying. If it had nice properties, like generating copying and being confluent, it could be used as a tool in proofs. However, we have already seen that:

$$\{\alpha = F(\alpha, \alpha)\} \to_c \{\alpha = F(\alpha', \alpha''),\ \alpha' = F(\alpha', \alpha),\ \alpha'' = F(\alpha, \alpha'')\}$$

This result cannot be obtained by sequential copying: Suppose it is possible. Because of symmetry this will have to be shown only for the case that α' is the first node to be added. We have two cases:

1. α'' is copied from α
   The inverse of the second step is $\alpha'' := \alpha$ so the result of the first step must be:

$$\{\alpha = F(\alpha', \alpha),\ \alpha' = F(\alpha', \alpha)\}$$

2. α'' is copied from α'
   The inverse of the second step is $\alpha'' := \alpha'$ so the result of the first step must be:

$$\{\alpha = F(\alpha', \alpha'),\ \alpha' = F(\alpha', \alpha)\}$$

Clearly both systems can be obtained from $\{\alpha = F(\alpha, \alpha)\}$ and they cannot be transformed into the required system by copying because the first argument of node α'' must be α and not α' or α'' which are the only possibilities.

1.2.3 μ-copying.

We will now briefly comment on copying as present in μ-calculus. For an introduction to μ-expressions and the missing definitions see [2].
A μ-step from one μ-expression to another translates to a copying step on the term graphs related with the μ-expressions. That copying step however might show a problem:
Take the μ-expression $\mu x.F(\mu y.G(x, H(y)))$. It translates to:

$$M = \{\alpha = F(\beta), \gamma = H(\beta), \beta = G(\alpha, \gamma)\}$$

It also transforms in a single μ-step to: $F(\mu y.G(\mu x.F(\mu z.G(x, H(z))), H(y)))$, which translates to:

$$M' = \{\alpha = F(\beta), \beta = G(\alpha', \gamma), \alpha' = F(\beta'), \beta' = G(\alpha', \gamma'), \gamma = H(\beta), \gamma' = H(\beta')\}$$

Clearly $M \rightarrow_c M'$. However this is not sequential. For the last step we have three options, we apply the inverse of those steps to M'.

$\gamma' := \gamma$

$$\{\alpha = F(\beta), \beta = G(\alpha', \gamma), \alpha' = F(\beta'), \beta' = G(\alpha', \gamma), \gamma = H(\beta), \gamma = H(\beta')\}$$

$\alpha' := \alpha$

$$\{\alpha = F(\beta), \gamma = H(\beta), \gamma' = H(\beta'), \beta = G(\alpha, \gamma), \alpha = F(\beta'), \beta' = G(\alpha, \gamma')\}$$

$\beta' := \beta$

$$\{\alpha = F(\beta), \beta = G(\alpha', \gamma), \gamma = H(\beta), \alpha' = F(\beta), \beta = G(\alpha', \gamma'), \gamma' = H(\beta)\}$$

Neither of these sets of equations are term graphs, so M' is a normal form of the inverse of sequential copying and therefore μ-copying is not sequential.

1.2.4 Sequential copying and flattening.

Copying and flattening commute, however sequentiality can be lost:

$$\begin{array}{ccc}
\{\alpha = F(G(\alpha))\} & \rightarrow_c & \{\alpha = F(G(\beta)), \beta = F(G(\alpha))\} \\
\downarrow_f & & \downarrow_f \\
\{\alpha = F(\alpha'), \alpha' = G(\alpha)\} & \rightarrow_c & \{\alpha = F(\alpha'), \alpha' = G(\beta), \beta = F(\beta'), \beta' = G(\alpha)\}
\end{array}$$

The top copying transformation is sequential, the bottom one is not.

1.2.5 Some properties of $\rightarrow_{1c}$.

The unwinding of a term graph to a possibly infinite tree is the limit of a sequential copying process:

1. Use Breadth First Search to traverse the tree.

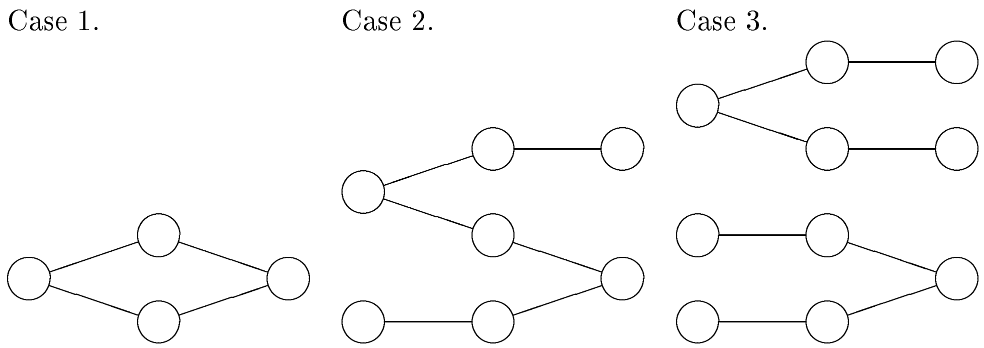
2. While the active node is of the form $\alpha = C[\beta]$, with β a reference to an already processed node (including α), copy γ from β and replace a single occurrence of β in $C[\beta]$ by γ.

3. Process the next node.

Observe that an ever increasing prefix of the term gets 'unshared'.

Theorem 1.2.1 *The inverse of sequential copying is $WCR^{\leq 1}$.*

Proof For the definition of WCR$^{\leq 1}$ see [7]. Assume g_1 and g_2 both transform into h in one step. We have $g_1 \underline{\leftrightarrow} g_2$ so $g_1 \wedge g_2$ exists. We know that $g_1 \wedge g_2$ copies to both g_1 and g_2. If we show that those copying steps are sequential steps, then we are finished. We will do so by counting the number of nodes of $g_1 \wedge g_2$.



Legenda: Each case has three columns, in the first column nodes of g_1 are shown, in the middle nodes from h and in the last nodes from g_2. The edges connect parent-child pairs from the copying steps. Nodes from g_1 with exactly one child in h, such that the parent of that node in h has exactly one child, are omitted.
The nodes of h and g_i are in $1-1$ correspondence except for two nodes in h that share the same one in g_i. For this there are three cases. (See picture above). We have $g_1 R_1 h R_2 g_2$, with R_1 and R_2 minimal bisimulations. If $R = R_1 \circ R_2$, then R is a bisimulation between g_1 and g_2. Now consider cases:

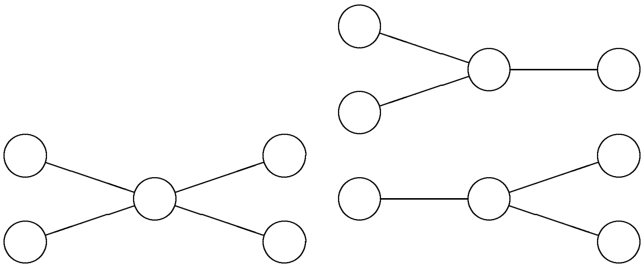
Case 1.
$g_1 \wedge g_2$ has the same number of nodes as g_1 and g_2, so $g_1 \equiv g_2$.

Case 2 & 3.
$g_1 \wedge g_2$ has at most one node less then g_i. So $g_1 \wedge g_2 \rightarrow_c g_i$ is indeed a sequential step.

Corollary 1.2.2 *Regarding term graphs modulo renaming, the equivalence relation induced by sequential copying is a proper subset of the one induced by general copying: Each equivalence class can be represented by a unique minimal (in number of nodes) term: the normal form of inverse sequential copying.*

For a finite acyclic term sequential copying suffices, because every bisimilar term sequentially copies to the unwinding of the term, which is unique.

Case 1. Case 2.



Legenda: Each case has three columns, in the first column nodes of g_1 are shown, in the middle nodes from h and in the last nodes from g_2. The edges connect parent-child pairs from the copying steps. Nodes from g_1 with exactly one child in h, such that the parent of that node in h has exactly one child, are omitted.
Sequential copying is not $WCR^{\leq 1}$. We have:

$$\{\alpha = F(\alpha, \alpha)\} \rightarrow_{1c} \{\alpha = F(\alpha', \alpha), \alpha' = F(\alpha, \alpha')\} = M$$

and

$$\{\alpha = F(\alpha, \alpha)\} \rightarrow_{1c} \{\alpha = F(\alpha, \alpha'), \alpha' = F(\alpha', \alpha)\} = N$$

We will construct a minimal bisimulation between M and N. It must include $(\alpha, \alpha) = \beta_0$ because roots must relate. It must include $(\alpha, \alpha') = \beta_1$ and $(\alpha', \alpha) = \beta_2$ because the arguments of related nodes must relate and finally it must include $(\alpha', \alpha') = \beta_3$, since it cannot be made larger. This must be all.
We have:

$$M \vee N \equiv \{\beta_0 = F(\beta_2, \beta_1), \beta_1 = F(\beta_3, \beta_0), \beta_2 = F(\beta_0, \beta_3), \beta_3 = F(\beta_1, \beta_2)\}$$

This is a minimal term with respect to sequential copying.
To me it is an open question whether sequential copying is confluent or not. The natural candidate in this case does not satisfy, and I have been unable to find any other.

Conjecture 1.2.3 *Sequential copying is non-confluent*

Finally we remark that it might be interesting to study the routines necessary to optimize a cyclic structure, think of dataflow programs, for least number of nodes.

Chapter 2

Proof systems for nested term graphs.

The proof systems in this chapter are inspired by the proof systems in [4] and [5].

2.1 Term graphs as Recursive Program Schemes

In a study of the proof system for RPS's in [4] I noticed that it is possible in that proof system to deduce that the associated programs of bisimilar graphs are equal. The proof system in [4] is not sound, but I believe it could be fixed with an added restriction to the induction rules used. I did not check if the forbidden constructs were used to obtain the main result of [4].

The rest of this section gives a more detailed account of the findings, but it can be skipped without impairing the understanding of the sequel.
A program is a list of length k of lines of the form:

$$F_i(X) \Leftarrow < term >$$

for $i = 1, \ldots, k$ where $< term >$ is a term over the finite signature $\mathcal{G} \cup \{F_i\}_{i=1}^{k}$ where each $g \in \mathcal{G}$ has a given arity and each F_i has arity 1. The only variable is X.
On these programs a notion of acceptability is defined, see [4].
Define the associated program $\mathcal{P}_G$ of a graph G with recursion variables $\{\alpha_i\}_{i=0}^{n}$ as:

$$\mathcal{P}_G = \{F_{\alpha_i}(X) \Leftarrow f_F(X, F_{\beta_1}(X), \ldots, F_{\beta_m}(X)\}_{i=0}^{n}$$

with

$$\alpha_i = F(\beta_1, \ldots, \beta_m) \in G$$

Notice that if α and β are the roots of G and H respectively, the following holds:

$$G \underline{\leftrightarrow} H \Leftrightarrow [\![F_\alpha(X)]\!] = [\![F_\beta(X)]\!] \text{ in the context of } \mathcal{P}_G \cup \mathcal{P}_H$$

Because the associated programs are acceptable we have:

$$G \underline{\leftrightarrow} H \Leftrightarrow \mathcal{P}_G, \mathcal{P}_H \vdash F_\alpha(X) \equiv F_\beta(X) \text{ is valid}$$

Because the proof system given is complete for this kind of equation, we can actually prove the assertion:

$$\mathcal{P}_G, \mathcal{P}_H \vdash F_\alpha(X) \equiv F_\beta(X)$$

To be able to use the proof system in [4] it is necessary to translate term graphs to RPS's, in the framework of [4] as indicated above. This is rather inconvenient. Therefore our first step will be to make a proof system that deals with term graphs directly. When that is done we can restrict the system removing the inequalities. Since global definitions are more difficult to work with than local definitions, when problems get larger, we then move on to a proof system for nested term graphs that extends standard equational logic.

2.2 The basic parts of proof systems.

In this section we deal with the parts of the proof systems that deal with abstract (in)equalities and plain logic. The precise structure of the terms that we are dealing with, is not yet important.

2.2.1 Definitions

Given some set of terms Terms, we now introduce the following basic concepts.

Definition 2.2.1

- An *atomic formula* can be:

an *equation*	$s = t$
an *inequality*	$s \leq t$

  with $s, t \in \mathsf{Terms}$.

- A *formula* is a set of atomic formulas. From now on Φ and Ψ will denote formulas. s, t and u are arbitrary terms.

- The semantics of a term t is $[\![t]\!]$, where $[\![.]\!] : \mathsf{Terms} \rightarrow \mathsf{Trees}$.

- $[\![\Phi]\!]$ is shorthand for $\{[\![s]\!] \sim [\![t]\!]\}_{s \sim t \in \Phi}$, where $\sim$ is $=$ or $\leq$.

- An *assertion* has the form

$$\Phi \vdash \Psi$$

- An assertion $\Phi \vdash \Psi$ is valid if the implication $(\bigwedge[\![\Phi]\!]) \Rightarrow (\bigwedge[\![\Psi]\!])$ holds for any substitution of the variables.

We now present our first proof system.

Proof system I: General base system.

(A1)	$\vdash t \leq t$	*reflexivity*
(A2)	$s \leq t, t \leq u \vdash s \leq u$	*transitivity*
(A3)	$s \leq t, t \leq s \vdash s = t$	*anti-symmetry*
(A4)	$s = t \vdash s \leq t, t \leq s$	*anti-symmetry*
(A5)	$\vdash t = t$	*reflexivity*
(A6)	$s = t \vdash t = s$	*symmetry*
(A7)	$s = t, t = u \vdash s = u$	*transitivity*
(A8)	$\Phi \vdash \Phi$	*quotation*
(R1)	$\dfrac{\Phi \vdash \Psi}{\Phi' \vdash \Psi'}$, for $\Phi \subset \Phi'$ and $\Psi \supset \Psi'$.	*inclusion*
(R2)	$\dfrac{\Phi_1 \vdash \Phi_2 \quad \Phi_2 \vdash \Phi_3}{\Phi_1 \vdash \Phi_3}$	*cut*
(R3)	$\dfrac{\Phi_1 \vdash \Psi_1 \quad \Phi_2 \vdash \Psi_2}{\Phi_1 \cup \Phi_2 \vdash \Psi_1 \cup \Psi_2}$	*conjunction*

2.2.2 Soundness

Theorem 2.2.1 (soundness of proof system I) *Every provable assertion is valid.*

Proof

A1:

$T\sigma \leq T\sigma$ holds for any substitution σ and any tree T. It certainly holds for $T = [\![t]\!]$, where t is a term. This means that:

$$\vdash t \leq t$$

is valid.

A2:

If for a substitution σ and terms s, t and u

$$[\![s]\!]\sigma \leq [\![t]\!]\sigma \text{ and } [\![t]\!]\sigma \leq [\![u]\!]\sigma$$

hold then by Proposition 1.1.2:

$$[\![s]\!]\sigma \leq [\![u]\!]\sigma$$

holds. So axiom A2 is valid.

A3,A4:

Because of Proposition 1.1.1:

$$[\![s]\!]\sigma \leq [\![t]\!]\sigma \text{ and } [\![t]\!]\sigma \leq [\![s]\!]\sigma \;\Leftrightarrow\; [\![s]\!]\sigma = [\![t]\!]\sigma$$

So

$$s \leq t, t \leq s \vdash s = t \text{ and } s = t \vdash s \leq t, t \leq s$$

are valid.

A8:

This axiom is absolutely trivially valid.

R1:

If $\Phi \vdash \Psi$ is valid then for any substitution σ:

$$\left(\bigwedge[\![\Phi]\!]\right)\sigma \Rightarrow \left(\bigwedge[\![\Psi]\!]\right)\sigma$$

Also if $\Gamma_2 \subset \Gamma_1$ then:

$$\left(\bigwedge[\![\Gamma_1]\!]\right)\sigma \Rightarrow \left(\bigwedge[\![\Gamma_2]\!]\right)\sigma$$

So:

$$\left(\bigwedge[\![\Phi']\!]\right)\sigma \Rightarrow \left[\left(\bigwedge[\![\Phi]\!]\right)\sigma \Rightarrow \left(\bigwedge[\![\Psi]\!]\right)\sigma \Rightarrow\right]\left(\bigwedge[\![\Psi']\!]\right)\sigma$$

R2:

If

$$\left(\bigwedge[\![\Phi_1]\!]\right)\sigma \Rightarrow \left(\bigwedge[\![\Phi_2]\!]\right)\sigma \text{ and } \left(\bigwedge[\![\Phi_2]\!]\right)\sigma \Rightarrow \left(\bigwedge[\![\Phi_3]\!]\right)\sigma$$

then

$$\left(\bigwedge[\![\Phi_1]\!]\right)\sigma \Rightarrow \left(\bigwedge[\![\Phi_3]\!]\right)\sigma$$

R3:

Observe that

$$\left(\bigwedge[\![\Phi_1 \cup \Phi_2]\!]\right)\sigma \Leftrightarrow \left(\bigwedge[\![\Phi_1]\!]\right)\sigma \bigwedge \left(\bigwedge[\![\Phi_2]\!]\right)\sigma$$

So if

$$\left(\bigwedge[\![\Phi_1]\!]\right)\sigma \Rightarrow \left(\bigwedge[\![\Psi_1]\!]\right)\sigma \text{ and } \left(\bigwedge[\![\Phi_2]\!]\right)\sigma \Rightarrow \left(\bigwedge[\![\Psi_2]\!]\right)\sigma$$

then

$$\left(\bigwedge[\![\Phi_1]\!]\right)\sigma \bigwedge \left(\bigwedge[\![\Phi_2]\!]\right)\sigma \Rightarrow \left(\bigwedge[\![\Psi_1]\!]\right)\sigma \bigwedge \left(\bigwedge[\![\Psi_2]\!]\right)\sigma$$

or

$$\left(\bigwedge[\![\Phi_1 \cup \Phi_2]\!]\right)\sigma \Rightarrow \left(\bigwedge[\![\Psi_1 \cup \Psi_2]\!]\right)\sigma$$

The other axioms can be derived from the axioms and rules so far proven sound and therefore are sound themselves:

A5:

1.	$\vdash t \leq t$	A1
2.	$t \leq t \vdash t = t$	A3
3.	$\vdash t = t$	R2, 1, 2

A6:

1.	$s = t \vdash s \leq t, t \leq s$	A4
2.	$t \leq s, s \leq t \vdash t = s$	A3
3.	$\vdash t = s$	R2, 1, 2

A7:

1.	$s = t \vdash s \leq t, t \leq s$	A4
2.	$t = u \vdash t \leq u, u \leq t$	A4
3.	$s = t, t = u \vdash s \leq t, t \leq s, t \leq u, u \leq t$	R3, 1, 2
4.	$s \leq t, t \leq u \vdash s \leq u$	A2
5.	$u \leq t, t \leq s \vdash u \leq s$	A2
6.	$s \leq t, t \leq u, u \leq t, t \leq s \vdash s \leq u, u \leq s$	R3, 4, 5
7.	$s \leq u, u \leq s \vdash s = u$	A3
8.	$s \leq t, t \leq u, u \leq t, t \leq s \vdash s = u$	R2, 6, 7
9.	$s = t, t = u \vdash s = u$	R2, 3, 8

□

2.2.3 Restriction to an equational proof system.

The base system can be syntactically restricted to disallow inequalities. The result is the following system:

Proof system II: Equational base system.

(A1)	$\vdash t = t$	*reflexivity*
(A2)	$s = t \vdash t = s$	*symmetry*
(A3)	$s = t, t = u \vdash s = u$	*transitivity*
(A4)	$\Phi \vdash \Phi$	*quotation*
(R1)	$\dfrac{\Phi \vdash \Psi}{\Phi' \vdash \Psi'}$, for $\Phi \subset \Phi'$ and $\Psi \supset \Psi'$.	*inclusion*
(R2)	$\dfrac{\Phi_1 \vdash \Phi_2 \quad \Phi_2 \vdash \Phi_3}{\Phi_1 \vdash \Phi_3}$	*cut*
(R3)	$\dfrac{\Phi_1 \vdash \Psi_1 \quad \Phi_2 \vdash \Psi_2}{\Phi_1 \cup \Phi_2 \vdash \Psi_1 \cup \Psi_2}$	*conjunction*

This system is sound, because it is a subset of the base system.

2.2.4 Parameterized proof systems.

When given a function $[\![.]\!]_{\mathcal{P}}$ that depends on the parameter $\mathcal{P}$ we can introduce that parameter into the systems.
A *parameterized assertion* has the form

$$\mathcal{P}, \Phi \vdash \Psi$$

The assertion is *valid* if the assertion

$$\Phi \vdash \Psi$$

is valid for the semantics $[\![.]\!]_{\mathcal{P}}$.
This gives us two more base systems.

Proof system III: Parameterized general base system.

(A1)	$\mathcal{P} \vdash t \leq t$	*reflexivity*
(A2)	$\mathcal{P}, s \leq t, t \leq u \vdash s \leq u$	*transitivity*
(A3)	$\mathcal{P}, s \leq t, t \leq s \vdash s = t$	*anti-symmetry*
(A4)	$\mathcal{P}, s = t \vdash s \leq t, t \leq s$	*anti-symmetry*
(A5)	$\mathcal{P} \vdash t = t$	*reflexivity*
(A6)	$\mathcal{P}, s = t \vdash t = s$	*symmetry*
(A7)	$\mathcal{P}, s = t, t = u \vdash s = u$	*transitivity*
(A8)	$\mathcal{P}, \Phi \vdash \Phi$	*quotation*
(R1)	$\dfrac{\mathcal{P}, \Phi \vdash \Psi}{\mathcal{P}, \Phi' \vdash \Psi'}$, for $\Phi \subset \Phi'$ and $\Psi \supset \Psi'$.	*inclusion*
(R2)	$\dfrac{\mathcal{P}, \Phi_1 \vdash \Phi_2 \quad \mathcal{P}, \Phi_2 \vdash \Phi_3}{\mathcal{P}, \Phi_1 \vdash \Phi_3}$	*cut*
(R3)	$\dfrac{\mathcal{P}, \Phi_1 \vdash \Psi_1 \quad \mathcal{P}, \Phi_2 \vdash \Psi_2}{\mathcal{P}, \Phi_1 \cup \Phi_2 \vdash \Psi_1 \cup \Psi_2}$	*conjunction*

Proof system IV: Parameterized equational base system.

(A1)	$\mathcal{P} \vdash t = t$	*reflexivity*
(A2)	$\mathcal{P}, s = t \vdash t = s$	*symmetry*
(A3)	$\mathcal{P}, s = t, t = u \vdash s = u$	*transitivity*
(A4)	$\mathcal{P}, \Phi \vdash \Phi$	*quotation*
(R1)	$\dfrac{\mathcal{P}, \Phi \vdash \Psi}{\mathcal{P}, \Phi' \vdash \Psi'}$, for $\Phi \subset \Phi'$ and $\Psi \supset \Psi'$.	*inclusion*
(R2)	$\dfrac{\mathcal{P}, \Phi_1 \vdash \Phi_2 \quad \mathcal{P}, \Phi_2 \vdash \Phi_3}{\mathcal{P}, \Phi_1 \vdash \Phi_3}$	*cut*
(R3)	$\dfrac{\mathcal{P}, \Phi_1 \vdash \Psi_1 \quad \mathcal{P}, \Phi_2 \vdash \Psi_2}{\mathcal{P}, \Phi_1 \cup \Phi_2 \vdash \Psi_1 \cup \Psi_2}$	*conjunction*

Both systems are sound because they are implementations of the non-parameterized versions. The parameter is at this point just syntactic noise.

2.3 A μ-calculus like proof system.

2.3.1 The proof system.

For a given set of variables $\mathcal{V}$ and a set of functions $\mathcal{F}$, including the special constant $\perp$, the undefined value, we first introduce some new concepts.

Definition 2.3.1

- Terms will be the set of simple terms over $\mathcal{V}$ and $\mathcal{F}$.

- Assertions are parameterized with programs.
- If the program is the empty set it may be omitted.

The proof system is an extension of the parameterized general base system (Proof system III). The new axioms and rules are:

Proof system V.

(A9)	$\mathcal{P} \vdash \perp \leq t$	*minimality*
(A10)	$\mathcal{P}, \{s_1 \leq t_1, \ldots, s_n \leq t_n\} \vdash f(s_1, \ldots, s_n) \leq f(t_1, \ldots, t_n)$	*monotonicity*
(A11)	$\mathcal{P} \vdash t = t\{s/\alpha\}$, if $\alpha = s \in \mathcal{P}$	*program*
(R4)	$\dfrac{\mathcal{P}, \Phi \vdash \Psi}{\mathcal{P}', \Phi \vdash \Psi}$, if $\mathcal{P} \subset \mathcal{P}'$.	*inclusion*
(R5)	$\dfrac{\mathcal{P} \cup \{\alpha = C[\beta], \beta = t\}, \Phi \vdash \Psi}{\mathcal{P} \cup \{\alpha = C[t], \beta = t\}, \Phi \vdash \Psi}$	*substitution*
(R6)	$\dfrac{\mathcal{P}, \Phi \vdash \Psi}{\mathcal{P}', \Phi \vdash \Psi}$, if $\mathcal{P} \supset \mathcal{P}'$ and no new free variables appear.	*garbage*
(R7)	$\dfrac{\mathcal{P}, \Phi \vdash \Psi\{\perp/\alpha_i\}_{i=1}^n \quad \mathcal{P}, \Phi, \Psi \vdash \Psi\{t_i/\alpha_i\}_{i=1}^n}{\mathcal{P} \cup \{\alpha_1 = t_1, \ldots, \alpha_n = t_n\}, \Phi \vdash \Psi}$, if no α_i occurs in $\mathcal{P}$ or Φ and no t_i is a variable.	*induction*

The proof system is given in full detail in Proof system IX on page 38.

Definition 2.3.2 $[\![t]\!]_{\mathcal{P}}$, the semantics of the term t relative to the program $\mathcal{P}$.

1. $[\![\alpha]\!]_{\mathcal{P}} = [\![t]\!]_{\mathcal{P}}$, if $\alpha = t \in \mathcal{P}$
2. $[\![\alpha]\!]_{\mathcal{P}} = \alpha$, if α undefined in $\mathcal{P}$.
3. $[\![f(t_1, \ldots, t_n)]\!]_{\mathcal{P}} = f([\![t_1]\!]_{\mathcal{P}}, \ldots, [\![t_n]\!]_{\mathcal{P}})$
4. $[\![\perp]\!]_{\mathcal{P}} = \perp$

Definition 2.3.3 The *flat* version of a program $\{\alpha^i = t_i\}_{i=1}^n$ is

$$\cup_{i=1}^n T_i \text{ where}$$

$$T_i = \left\{ \alpha_w^i = f(X_1, \ldots, X_{arity(f)}) \;\middle|\; wf \in t_i, f \in \mathcal{F}, X_j = \left\{ \begin{array}{ll} \alpha_\epsilon^k & \text{if } wj\alpha^k \in t_i \\ \alpha & \text{if } wj\alpha \in t_i \text{ and } \alpha \text{ free} \\ \alpha_{wj}^i & \text{otherwise} \end{array} \right. \right\}$$

looking at t_i as a tree.

Example 2.3.1 The flat version of:

$$\{\alpha = f(\beta, \gamma), \beta = g(h(\alpha), \beta)\}$$

is:

$$\{\alpha_\epsilon = f(\beta_\epsilon, \gamma), \beta_\epsilon = g(\beta_1, \beta_\epsilon), \beta_1 = h(\alpha_\epsilon)\}$$

2.3.2 Soundness

Theorem 2.3.1 (soundness of proof system V) *Every provable assertion is valid.*

Proof

A9:
Since $[\![\bot]\!] = \bot$ and $\bot \leq T$ for arbitrary T we have that $\bot = \bot\ \sigma \leq [\![t]\!]\sigma$ so (A9) is valid.

A10:
The assertion is valid if the implication

$$[\![s_1]\!]\sigma \leq [\![t_1]\!]\sigma, \ldots, [\![s_n]\!]\sigma \leq [\![t_n]\!]\sigma \Rightarrow [\![f(\vec{s})]\!]\sigma \leq [\![f(\vec{c})]\!]\sigma$$

holds for all σ. It holds and it even is an equivalence.

A11:
Observe that $[\![t]\!]_{\mathcal{P}} = [\![t\{s/\alpha\}]\!]_{\mathcal{P}}$, if $\alpha = s \in \mathcal{P}$. The rest is routine.

R4:
If

$$[\![\Phi]\!]_{\mathcal{P}} \Rightarrow [\![\Psi]\!]_{\mathcal{P}}$$

then

$$([\![\Phi]\!]_{\mathcal{P}} \Rightarrow [\![\Psi]\!]_{\mathcal{P}})\,\{T/\alpha\}$$

for any free α and any T.
Take $T = [\![\alpha]\!]_{\mathcal{P}\cup\{\alpha=t\}}$ then we have:

$$[\![\Phi]\!]_{\mathcal{P}\cup\{\alpha=t\}} \Rightarrow [\![\Psi]\!]_{\mathcal{P}\cup\{\alpha=t\}}$$

Repeated use gives:

$$[\![\Phi]\!]_{\mathcal{P}'} \Rightarrow [\![\Psi]\!]_{\mathcal{P}'}$$

R5:
Actually the functions $[\![.]\!]_{\mathcal{P}\cup\{\alpha=C[\beta],\beta=t\}}$ and $[\![.]\!]_{\mathcal{P}\cup\{\alpha=C[t],\beta=t\}}$ are the same.

R6:
Validity doesn't depend on the definition of inaccessible variables.

R7:
Put $\sigma = \{t_i/\alpha_i\}_{i=1}^{n}$ and $\sigma_\bot = \{\bot/\alpha_i\}_{i=1}^{n}$.
Claim:

$$\mathcal{P}, \Phi \vdash \Psi\,(\sigma^n\sigma_\bot)\,,\ \text{for all } n \in \mathbb{N}$$

Proof of the claim: for $n = 0$ it is a copy of the top left assertion. Assume the assertion is true for a certain n then, since

$$\mathcal{P}, \Phi, \Psi \vdash \Psi\sigma$$

is valid, we have:

$$(\mathcal{P}, \Phi, \Psi \vdash \Psi\sigma)\,(\sigma^n\sigma_\bot)$$

is valid, which means that:

$$\mathcal{P}, \Phi, \Psi\left(\sigma^n \sigma_\perp\right) \vdash \Psi\left(\sigma^{n+1}\sigma_\perp\right)$$

is valid. By assumption:

$$\mathcal{P}, \Phi \vdash \Psi\left(\sigma^n \sigma_\perp\right)$$

is valid so:

$$\mathcal{P}, \Phi \vdash \Psi\left(\sigma^{n+1}\sigma_\perp\right)$$

is valid, which proves the claim.
If we put $\mathcal{P}' = \mathcal{P} \cup \{\alpha_1 = t_1, \ldots, \alpha_n = t_n\}$ we want:

$$\mathcal{P}', \Phi \vdash \Psi$$

Here we remark that:

$$\lim_{n\to\infty} [\![\Psi\sigma^n\sigma_\perp]\!]_{\mathcal{P}} = [\![\Psi]\!]_{\mathcal{P}'}$$

given a substitution τ such that $[\![\Phi]\!]_{\mathcal{P}}\tau \equiv [\![\Phi]\!]_{\mathcal{P}'}\tau$ is true, we have that for all n $[\![\Psi\sigma^n\sigma_\perp]\!]\sigma$ is true. This last construct consists of a number of inequalities $S_n \leq T_n$ (equalities also fall under this assumption via Proposition 1.1.1) such that

$$\lim_{n\to\infty} S_n = S \text{ and } \lim_{n\to\infty} T_n = T$$

If we can prove that $S \leq T$ we are done. By definition of $\leq$ we must prove that $S_\perp \subset T_\perp$. We will do that by proving that for all $n \in \mathbb{N}$ we have that $S_\perp{\uparrow_n} \subset T_\perp{\uparrow_n}$. Let n be given. Since $S_i \to S$ we know that for a certain N

$$\forall j > N : S_j{\uparrow_n} = S{\uparrow_n}$$

Choose a certain $j > N$ then since $(S{\uparrow_n})_\perp = S_\perp{\uparrow_n}$ we know that:

$$S_\perp{\uparrow_n} = (S{\uparrow_j})_\perp \subset (T{\uparrow_j})_\perp = T_\perp{\uparrow_n}$$

the middle inclusion holds because of $S_j \leq T_j$. So R7 is sound.

□

2.3.3 Completeness

Theorem 2.3.2 (completeness of proof system V) *The proof system is complete with respect to assertions of the form:*

$$\mathcal{P} \vdash s = t$$

This is a simple corollary from Theorem 2.4.2. That theorem will be proven in the next section.

2.4 A proof system for term graphs.

2.4.1 The proof system

The set of terms and the semantics relative to a program $\mathcal{P}$, $[\![.]\!]_{\mathcal{P}}$ are the same as in the previous section.
Extend the equational base system with programs for parameters (Proof system IV) with:

Proof system VI.

(A5)	$\mathcal{P}, \{s_1 = t_1, \ldots, s_n = t_n\} \vdash f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)$	*congruence*
(A6)	$\mathcal{P} \vdash t = t\{s/\alpha\}$, if $\alpha = s \in \mathcal{P}$	*program*
(R4)	$\dfrac{\mathcal{P}, \Phi \vdash \Psi}{\mathcal{P}', \Phi \vdash \Psi}$, if $\mathcal{P} \subset \mathcal{P}'$.	*inclusion*
(R5)	$\dfrac{\mathcal{P} \cup \{\alpha = C[\beta], \beta = t\}, \Phi \vdash \Psi}{\mathcal{P} \cup \{\alpha = C[t], \beta = t\}, \Phi \vdash \Psi}$	*substitution*
(R6)	$\dfrac{\mathcal{P}, \Phi \vdash \Psi}{\mathcal{P}', \Phi \vdash \Psi}$, if $\mathcal{P} \supset \mathcal{P}'$ and no new free variables appear.	*garbage*
(R7)	$\dfrac{\mathcal{P}, \Phi \vdash \Psi\{\perp/\alpha_i\}_{i=1}^n \quad \mathcal{P}, \Phi, \Psi \vdash \Psi\{t_i/\alpha_i\}_{i=1}^n}{\mathcal{P} \cup \{\alpha_1 = t_1, \ldots, \alpha_n = t_n\}, \Phi \vdash \Psi}$, if no α_i occurs in $\mathcal{P}$ or Φ and no t_i is a variable.	*induction*

The proof system is given in full detail in Proof system X on page 39.

Example 2.4.1 We will give a formal deduction of

$$\alpha = F(\alpha, \alpha), \beta = F(\beta, \gamma), \gamma = F(\gamma, \beta) \vdash \alpha = \beta$$

1.	$\vdash \perp = \perp$	A1
2.	$\vdash \{\alpha = \beta, \alpha = \gamma\}\{\perp/\alpha, \perp/\beta, \perp/\gamma\}$	syntax, 1
3.	$\alpha = \beta, \alpha = \gamma \vdash F(\alpha, \alpha) = F(\beta, \gamma)$	A5
4.	$\alpha = \beta, \alpha = \gamma \vdash F(\alpha, \alpha) = F(\gamma, \beta)$	A5
5.	$\alpha = \beta, \alpha = \gamma \vdash F(\alpha, \alpha) = F(\beta, \gamma), F(\alpha, \alpha) = F(\gamma, \beta)$	R3, 3, 4
6.	$\alpha = \beta, \alpha = \gamma \vdash \{\alpha = \beta, \alpha = \gamma\}\{F(\alpha, \alpha)/\alpha, F(\beta, \gamma)/\beta, F(\gamma, \beta)/\gamma\}$	syntax, 6
7.	$\alpha = F(\alpha, \alpha), \beta = F(\beta, \gamma), \gamma = F(\gamma, \beta) \vdash \alpha = \beta, \alpha = \gamma$	R7, 2, 6
8.	$\alpha = F(\alpha, \alpha), \beta = F(\beta, \gamma), \gamma = F(\gamma, \beta) \vdash \alpha = \beta$	R1, 7

2.4.2 Soundness

Theorem 2.4.1 (soundness of proof system VI) *Every provable assertion is valid.*

Proof The system is actually a subset of the first one. For (A6), (R4), (R6) and (R1) this property can be seen immediately. For (A5) we need to do some work.
For $i = 0, \ldots, n$ put

$$\Phi_i = s_1 \leq t_1, t_1 \leq s_1, \ldots, s_i \leq t_i, t_i \leq s_i, s_{i+1} = t_{i+1}, \ldots, s_n = t_n$$

Expansion gives for $i = 1, \ldots, n$

$$s_i = t_i \vdash s_i \leq t_i, t_i \leq s_i$$

together with quotation and conjunction this gives

$$\Phi_{i-1} \vdash \Phi_i$$

Apply the cut rule n times to obtain:

$$\Phi_0 \vdash \Phi_n$$

We have

$$\Phi_n \vdash s_1 \leq t_1, \ldots, s_n \leq t_n \text{ and } \Phi_n \vdash t_1 \leq s_1, \ldots, t_n \leq s_n$$

and

$$s_1 \leq t_1, \ldots, s_n \leq t_n \vdash f(s_1, \ldots s_n) \leq f(t_1, \ldots t_n)$$

and

$$s_1 \leq t_1, \ldots, s_n \leq t_n \vdash f(t_1, \ldots t_n) \leq f(s_1, \ldots s_n)$$

So we have:

$$\Phi_0 \vdash f(s_1, \ldots s_n) = f(t_1, \ldots t_n)$$

□

2.4.3 Completeness

In this section we prove completeness, with respect to assertions of the form $\mathcal{P} \vdash \Phi$. Completeness for the general case of assertions of the form $\mathcal{P}, \Phi \vdash \Psi$ is probably possible, though one will need to extend the proof system.

Theorem 2.4.2 (completeness of proof system VI) *If the assertion $\mathcal{P} \vdash \Phi$ is valid then it is deducible in the proof system.*

To prove this we need several propositions.

Proposition 2.4.3 *If the assertion $\mathcal{P} \vdash \alpha = \beta$ is valid and $\mathcal{P}$ is flat, then it is deducible.*

Proof Assume $\mathcal{P} = \{\alpha_i = f_i(\alpha_{i,1}, \ldots, \alpha_{i,n_i})\}_{i=1}^n$. Since $[\![\alpha]\!]_\mathcal{P} = [\![\alpha]\!]_{\mathcal{P}@\alpha}$ and $[\![\beta]\!]_\mathcal{P} = [\![\beta]\!]_{\mathcal{P}@\beta}$, it follows that the unwindings of the term graphs $\mathcal{P}@\alpha$ and $\mathcal{P}@\beta$ are the same. This means that there is a bisimulation R between them. Put

$$\Phi = \{\gamma_1 = \gamma_2 | (\gamma_1, \gamma_2) \in R\}$$

Because $\vdash \perp = \perp$ is deducible, we can deduce:

$$\vdash \Phi\{\perp / \alpha_i\}_{i=1}^n$$

For each equation $\alpha_i = \alpha_j$ in Φ we know, because of the bisimulation, that $f_i \equiv f_j$ and that $n_i = n_j$ and for all $k = 1, \ldots, n_i$ we know that $(\alpha_{i,k}, \alpha_{j,k}) \in R$ which means that $\alpha_{i,k} = \alpha_{j,k} \in \Phi$. So $\Phi \vdash \alpha_i = \alpha_j\{f_i(\alpha_{i,1}, \ldots, \alpha_{i,n_i})/\alpha_i\}_{i=1}^n$ is deducible. So:

$$\Phi \vdash \Phi\{f_i(\alpha_{i,1}, \ldots, \alpha_{i,n_i})/\alpha_i\}_{i=1}^n$$

is deducible. Which means that by induction

$$\mathcal{P} \vdash \Phi$$

is deducible and since $\alpha = \beta \in \Phi$ also

$$\mathcal{P} \vdash \alpha = \beta$$

is deducible.

□

Proposition 2.4.4 *If $\mathcal{Q}$ is the flattened version of a program $\mathcal{P}$ then*

$$\mathcal{Q} \cup \mathcal{P} \vdash \{\alpha = \alpha_\epsilon\}_{\alpha \in def(\mathcal{P})}$$

is deducible.

Proof We know that $\vdash \perp = \perp$ is deducible. So:

$$\vdash \left[\{\alpha^w = \alpha_w\}_{\alpha_w \in def(\mathcal{Q})}\right] \{\perp / \alpha_w, \perp / \alpha^w\}_{\alpha_w \in def(\mathcal{Q})}$$

is deducible. It is also obvious that

$$\{\alpha^w = \alpha_w\}_{\alpha_w \in def(\mathcal{Q})} \vdash \left[\{\alpha = \alpha_\epsilon\}_{\alpha \in def(\mathcal{P})}\right] \{t_w/\alpha_w, t_w/\alpha^w\}_{\alpha_w \in def(\mathcal{Q})}$$

is deducible by symmetry of $=$. The induction rule gives that:

$$\mathcal{Q} \cup \{\alpha^w = t_w\}_{\alpha_w \in def(\mathcal{Q})} \vdash \{\alpha^w = \alpha_w\}_{\alpha_w \in def(\mathcal{Q})}$$

From this it is deducible that:

$$\mathcal{Q} \cup \{\alpha^w = t_w\}_{\alpha_w \in def(\mathcal{Q})} \vdash \{\alpha^\epsilon = \alpha_\epsilon\}_{\alpha \in def(\mathcal{P})}$$

Apply R5 as long as a variable with a superscript other than ϵ occurs in the definition of a variable with a superscript ϵ. This is a terminating process. When finished drop the defining equations of variables with a non-ϵ superscript, by rule R6:

$$\mathcal{Q} \cup \{\alpha^\epsilon = t\}_{\alpha = t \in \mathcal{P}} \vdash \{\alpha^\epsilon = \alpha_\epsilon\}_{\alpha \in def(\mathcal{P})}$$

Removing the superscripts yields the result we were looking for.

Proposition 2.4.5 *If the assertion $\mathcal{P} \vdash \alpha = \beta$ is valid, then it is deducible.*

Proof If $\mathcal{Q}$ is the flattened version of $\mathcal{P}$ then $\mathcal{Q} \vdash \alpha_\epsilon = \beta_\epsilon$ is also valid. And so:

1.	$\mathcal{Q} \vdash \alpha_\epsilon = \beta_\epsilon$	deducible by Proposition 2.4.3
2.	$\mathcal{Q} \cup \mathcal{P} \vdash \alpha_\epsilon = \beta_\epsilon$	R1,1
3.	$\mathcal{Q} \cup \mathcal{P} \vdash \{\alpha = \alpha_\epsilon\}_{\alpha \in def(\mathcal{P})}$	deducible by Proposition 2.4.4
4.	$\mathcal{Q} \cup \mathcal{P} \vdash \alpha = \beta$	equational reasoning,2,3
5.	$\mathcal{P} \vdash \alpha = \beta$	R6,4

□

Proposition 2.4.6 *If the assertion $\mathcal{P} \vdash s = t$ is valid then it is deducible in the system.*

Proof If s and t are guarded terms then w.l.o.g. assume $\{\alpha, \beta\} \cap def(\mathcal{P}) = \emptyset$. We can deduce:

1.	$\mathcal{P} \vdash s = t$	given
2.	$\mathcal{P} \cup \{\alpha = s, \beta = t\} \vdash s = t$	R1
3.	$\mathcal{P} \cup \{\alpha = s, \beta = t\} \vdash \alpha = s$	A6
4.	$\mathcal{P} \cup \{\alpha = s, \beta = t\} \vdash \beta = t$	A6
5.	...	
6.	$\mathcal{P} \cup \{\alpha = s, \beta = t\} \vdash \alpha = \beta$	exercise

by assumption the first line is valid, so the last is also, and 2.4.5 gives a deduction for that line. We then make the following deduction:

1.	$\mathcal{P} \cup \{\alpha = s, \beta = t\} \vdash \alpha = \beta$	Given
2.	$\mathcal{P} \cup \{\alpha = s, \beta = t\} \vdash \alpha = s$	A6
3.	$\mathcal{P} \cup \{\alpha = s, \beta = t\} \vdash \beta = t$	A6
4.	$\mathcal{P} \cup \{\alpha = s, \beta = t\} \vdash s = t$	in several steps
5.	$\mathcal{P} \vdash s = t$	R6,4

The first line can be deduced, so also the last. If s or t is a variable it is even easier.

□

Proof of Theorem 2.4.2 Given a valid assertion

$$\mathcal{P} \vdash \Phi$$

we may write $\Phi \equiv s_1 = t_1, \ldots, s_n = t_n$. By the definition of validity we know that for $i = 1, \ldots, n$

$$\mathcal{P} \vdash s_i = t_i$$

is valid and by Proposition 2.4.6 we have deductions for them. Applying R3 $n-1$ times we finally obtain that:

$$\mathcal{P} \vdash s_1 = t_1, \ldots, s_n = t_n \equiv \mathcal{P} \vdash \Phi$$

□

2.5 A proof system for nested term graphs.

2.5.1 The proof system.

Define $\bullet \equiv \perp$. $Def(E)$ is the set of variables that occur as a left-hand side of the environment E. $Var(E)$ is the set of variables that just occurs inside the environment E.
Extend the equational base system (proof system II) with:

Proof system VII.

(A5)	$\vdash \langle t \vert \alpha = \alpha, E\rangle = \langle t \vert \alpha = \bullet, E\rangle$	*black hole*
(A6)	$\vdash \langle t \vert \alpha = \beta, E\rangle = \langle t \vert E\rangle\{\beta/\alpha\}$, where α and β are different variables	*renaming*
(A7)	$\vdash \langle \alpha \vert \alpha = t, E\rangle = \langle t \vert \alpha = t, E\rangle$	*substitution*
(A8)	$\vdash s = t$, if $s \to_\alpha t$	*alpha conversion*
(A9)	$\langle s \vert E\rangle = \langle s \vert F\rangle$, if $E = F$ modulo garbage collection	*garbage collection*
(A10)	$\vdash \langle f(s_1, \ldots, s_n) \vert E\rangle = f(\langle s_1 \vert E\rangle, \ldots, \langle s_n \vert E\rangle)$	*unwinding*
(A11)	$\vdash \langle t \vert \rangle = t$	*box-removal*
(A12)	$\vdash \langle\langle s \vert E\rangle \vert F\rangle = \langle s \vert E, F\rangle$, if $Def(E) \cap Var(F) = \emptyset$.	*merging*
(A13)	$s = t \vdash C[s] = C[t]$, for any context $C[.]$ that does not capture free variables from s or t.	*context*
(R4)	$\dfrac{\vdash s' = t'}{\vec{s} = \vec{t} \vdash s'\{\vec{s}/\vec{\alpha}\} = t'\{\vec{t}/\vec{\alpha}\}}$	*substitution*
(R5)	$\dfrac{\vdash s = t}{\vdash C[s] = C[t]}$, for any context $C[.]$	*context*
(R6)	$[s_i = t_i(\vec{s})]_{i=1}^n \vdash s_1 = \langle \alpha_1 \vert [\alpha_i = t_i(\vec{\alpha})]_{i=1}^n\rangle$, if each $t_i(\vec{\beta})$ is guarded and all α_i are fresh.	*folding*
(R7)	$\dfrac{\Phi \vdash \Psi\{s_i/\alpha_i\}_{i=1}^n \quad \Phi, \Psi \vdash \Psi\{t_i/\alpha_i\}_{i=1}^n}{\Phi \vdash \Psi^{\{\alpha_i = t_i\}_{i=1}^n}}$, for any s_i, if no α_i is free in Φ and each t_i is guarded.	*induction*

The system is given in full detail in proof system XI on page 40.

Definition 2.5.1 Semantics ($[\![.]\!]$) of nested term graphs.

1. $[\![f(s_1, \ldots, s_n)]\!] = f([\![s_1]\!], \ldots, [\![s_n]\!])$
2. $[\![\alpha]\!] = \alpha$
3. $[\![\langle f(s_1, \ldots, s_n) \vert E\rangle]\!] = f([\![\langle s_1 \vert E\rangle]\!], \ldots, [\![\langle s_n \vert E\rangle]\!])$
4. $[\![\langle \alpha \vert E\rangle]\!] = [\![\langle T \vert E\rangle]\!]$. To obtain T begin a recursive sequence. The first element is α. If the last element is not a variable that is defined in E then the process stops, otherwise the next element is $[\![s]\!]$ with $\beta = s \in E$ and we repeat the extension process. T is the last element of the sequence if it is finite and $\bullet$ otherwise.
5. $[\![\langle\langle \alpha \vert E\rangle \vert F\rangle]\!] = [\![\langle [\![\langle \alpha \vert E\rangle]\!] \vert F\rangle]\!]$

Strictly speaking $\langle [\![t]\!] \vert E\rangle$ is not a NTG, but there is no harm done in pretending it is. In the construction of the sequence we may use $[\![.]\!]$ because it is applied to a NTG with a lower degree of nesting.

Proposition 2.5.1 *For any nested term graph t and any context $C[.]$ the following holds:*

$$[\![C[t]]\!] = [\![C[[\![t]\!]]]\!]$$

Proof Observe that $[\![s]\!] = [\![[\![s]\!]]\!]$.

$\square$

Definition 2.5.2

- A NTG is *graph-like* if it's of the form $\langle \alpha_1 | \vec{\alpha} = \vec{t} \rangle$ for a vector of terms (without nesting constructs!) $\vec{t}$.
- A NTG is *flat* if it is a variable with an exponent that is a flat program.

2.5.2 Soundness

Definition 2.5.3 For an environment E the substitution σ_E is defined by:

$$\sigma_E(\alpha) = [\![\alpha^E]\!]$$

If τ is a substitution and $V \subset \mathcal{V}$ then

$$\tau{\uparrow_V}(\alpha) = \begin{cases} \tau(\alpha) & , \alpha \in V \\ \alpha & , \text{otherwise} \end{cases}$$

V^c will be short for $\{v \in \mathcal{V} | v \notin V\}$.

Proposition 2.5.2 *For any nested term graph s*

$$[\![s^E]\!] = [\![s]\!]\sigma_E$$

Proof With induction to the construction of s:

$$[\![\alpha^E]\!] = \sigma_E(\alpha) = (\alpha)\sigma_E$$

$$[\![f(s_1, \ldots, s_n)^E]\!] = f([\![s_1^E]\!], \ldots, [\![s_n^E]\!]) = f([\![s_1]\!]\sigma_E, \ldots, [\![s_n]\!]\sigma_E) = [\![f(s_1, \ldots, s_n)]\!]\sigma_E$$

$$[\![\langle t|F\rangle^E]\!] = [\![[\![t^F]\!]^E]\!] = [\![t^F]\!]\sigma_E$$

$\square$

Theorem 2.5.3 (soundness of proof system VII) *If an assertion is provable then it is valid.*

Proof

(A5)

When looking at the definition of $[\![.]\!]$ it is clear that the only place where the effect of a different environment can be felt is where a variable is substituted. It is so that:

$$[\![\langle \beta | \alpha = \alpha, E\rangle]\!] = [\![\langle \beta | \alpha = \bullet, E\rangle]\!]$$

because if α does not occur in the sequence there is no difference and if α occurs in the first case we get an infinite sequence and in the second a sequence ending with $\bullet$, so in both cases $T = \bullet$. We now have that $[\![\langle t|\alpha = \alpha, E\rangle C]\!] = [\![\langle t|\alpha = \bullet, E\rangle C]\!]$ and so:

$$\vdash \langle t|\alpha = \alpha, E\rangle = \langle t|\alpha = \bullet, E\rangle$$

(A6)

$[\![\langle\gamma|\alpha=\beta,E\rangle]\!] = [\![\langle\gamma|E\rangle\{\beta/\alpha\}]\!]$, because if α does not occur in the sequence of γ there is no difference at all, and if α occurs then the sequence in the second case is derived from that is the first by erasing all occurrences of α. Since α cannot be the last symbol and an infinite sequence stays infinite, because in the first case every α is immediately followed by a β, so should one delete infinitely many α's one will still have infinitely many β's. Therefore (A6) is sound.

(A7)

For the case that $t \equiv \alpha$, we can use axiom A2.

For the case that $t \equiv \beta \neq \alpha$, we must prove that

$$\vdash \langle\alpha|\alpha=\beta,E\rangle = \langle\beta|\alpha=\beta,E\rangle$$

is valid, using axiom A6 we can deduce that

$$\vdash \langle\alpha|\alpha=\beta,E\rangle = \langle\alpha|E\rangle\{\beta/\alpha\}(\equiv \langle\beta|E\rangle\{\beta/\alpha\}), \langle\beta|\alpha=\beta,E\rangle = \langle\beta|E\rangle\{\beta/\alpha\}$$

from which the requested equality can be deduced.

For the case that t is not a variable the definition of $[\![.]\!]$ gives that:

$$[\![\langle\alpha|\alpha=t,E\rangle]\!] = [\![\langle[\![t]\!]|\alpha=t,E\rangle]\!] = [\![\langle t|\alpha=t,E\rangle]\!]$$

So for any t:

$$\vdash \langle\alpha|\alpha=t,E\rangle = \langle t|\alpha=t,E\rangle$$

is valid.

(A8)

The definition of $[\![.]\!]$ does not depend on the names of defined variables. Therefore $[\![s]\!] = [\![t]\!]$, if $s \to_\alpha t$.

(A9)

An inaccessible variable will never occur in the sequence of an accessible variable, so one can freely add and remove inaccessible defining equations.

(A10)

By definition of $[\![.]\!]$ the semantics of left- and right-hand sides are equal.

(A11)

If $E = \emptyset$ then $\sigma_E \equiv \mathsf{id}$ so $[\![t^E]\!] = [\![t]\!]\sigma_E = [\![t]\!]$.

(A12)

$$[\![\langle\langle s|E\rangle|F\rangle]\!] = [\![\langle s|E\rangle]\!]\sigma_F = [\![s]\!]\sigma_E\sigma_F$$
$$[\![\langle s|E,F\rangle]\!] = [\![s]\!]\sigma_{E,F}$$

So what we want is that for all $\alpha \in \mathcal{V}$

$$\sigma_{E,F}(\alpha) = \sigma_F(\sigma_E(\alpha))$$

holds. For $\alpha \notin Def(E)$ this is trivial. So assume $\alpha \in Def(E)$. We then have:

$$\sigma_{E,F}(\alpha) = [\![\langle\alpha|E,F\rangle]\!] = T([\![\vec{\alpha}^{E,F}]\!])$$

$$\sigma_F(\sigma_E(\alpha)) = [\![\langle\alpha|E\rangle]\!]\sigma_F = T([\![\vec{\alpha}^{E}]\!])\sigma_F = T([\![\vec{\alpha}^{E}]\!]\sigma_F)$$

The T is the same for both cases and is obtained by unwinding *inside* the E-part of the environment. The variables that are left are variables whose list touches the F part (from which it cannot escape.) Observe that for a variable $\beta \notin Def(E)$ we have that $[\![\langle\beta|E,F\rangle]\!]$ is the same as $[\![\langle\beta|F\rangle]\!]$ and that $[\![\langle\beta|E\rangle]\!]\sigma_F = [\![\langle\beta|F\rangle]\!]$.

(A13)

W.l.o.g. assume that α does not occur in $C[.]$ or s or t. Clearly $[\![C[\alpha]]\!] = [\![C[\alpha]]\!]$. If $[\![s]\!]\sigma = [\![t]\!]\sigma$ then also $[\![C[\alpha]]\!]\{[\![s]\!]\sigma/\alpha\} = [\![C[\alpha]]\!]\{[\![t]\!]\sigma/\alpha\}$. Because of the noncaptive nature $[\![C[\alpha]]\!]\{[\![s]\!]\sigma/\alpha\} = [\![C[s]]\!]\sigma$ and $[\![C[\alpha]]\!]\{[\![t]\!]\sigma/\alpha\} = [\![C[t]]\!]\sigma$. So: $[\![C[s]]\!]\sigma = [\![C[t]]\!]\sigma$.

(R4)

If $\vdash s' = t'$ is valid then

$$[\![s']\!] = [\![t']\!]$$

Put

$$\sigma_s(\alpha_i) = [\![s_i]\!] \text{ and } \sigma_t(\alpha_i) = [\![t_i]\!]$$

then

$$[\![s'\{\vec{s}/\vec{\alpha}\}]\!] = [\![s']\!]\sigma_s \text{ and } [\![t'\{\vec{t}/\vec{\alpha}\}]\!] = [\![t']\!]\sigma_t$$

If for a certain τ, and $i = 1, \ldots, n$:

$$[\![s_i]\!]\tau = [\![t_i]\!]\tau$$

then $\sigma_t\tau \equiv \sigma_t\tau$ and therefore:

$$[\![s'\{\vec{s}/\vec{\alpha}\}]\!]\tau = [\![t'\{\vec{t}/\vec{\alpha}\}]\!]\tau$$

(R5)

Observe that:

$$[\![C[s]]\!] = [\![C[[\![s]\!]]]\!] = [\![C[[\![t]\!]]]\!] = [\![C[t]]\!]$$

(R6)

Folding is a special case of induction. Given that:

$$[s_i = t_i(s_1, \ldots, s_n)]_{i=1}^{n} = \Phi$$

with all t_i guarded, apply induction to

$$\Psi = s_1 = \alpha_1, \ldots, s_n = \alpha_n$$

It is trivial that

$$\Phi \vdash \Psi\{\vec{s}/\vec{\alpha}\}$$

Also:

$$\Phi, \Psi \vdash \vec{\alpha} = \vec{s} = \vec{t}(\vec{s}) = \vec{t}(\vec{\alpha})$$

So

$$\Phi, \Psi \vdash \Psi\{t(\vec{\alpha})/\vec{\alpha}\}$$

Induction yields ($E = \{\vec{\alpha} = \vec{t}(\vec{\alpha})\}$):

$$\Phi \vdash \vec{\alpha}^E = \vec{s}^E = \vec{s}$$

So in particular:

$$\Phi \vdash s_1 = \langle \alpha_1 | E \rangle$$

□

(R7)

Put $E = \{\vec{s}/\vec{\alpha}\}$ and $F = \{\vec{t}/\vec{\alpha}\}$.

Given that $\Phi \vdash \Psi E$ and $\Phi, \Psi \vdash \Psi F$ hold and given a substitution σ such that $[\![\Phi]\!]\sigma$ is true, $[\![\Phi F^n E]\!]\sigma = [\![\Phi]\!] F^n E\sigma$ also holds, because $\Phi E = \Phi F = \Phi$. The first assumption yields that $[\![\Psi E]\!]\sigma = [\![\Psi]\!] E\sigma$ holds. If $[\![\Psi]\!] F^n E\sigma$ holds the second assumption yields that $[\![\Psi F]\!] F^n E\sigma = [\![\Psi]\!] F^{n+1} E\sigma$ holds. So for any n $[\![\Psi F]\!] F^n E\sigma$ holds, which means that $[\![\Psi^{\{\vec{\alpha}=\vec{t}\}}]\!]\sigma$ holds, because:

$$\begin{array}{ccc} [\![s]\!] F^n E\sigma & = & [\![t]\!] F^n E\sigma \\ \downarrow & & \downarrow \\ [\![s^{\{\vec{\alpha}=\vec{t}\}}]\!]\sigma & = & [\![t^{\{\vec{\alpha}=\vec{t}\}}]\!]\sigma \end{array}$$

□

2.5.3 Completeness

Proposition 2.5.4 *Each term is provably equal to a term with guarded defining equations.*

Proof With induction to the nesting of the term. A simple term has no defining equations, so they are all guarded. Given a term t^E assume that defining equations inside a definition in E have guarded right-hand sides. For unguarded defining equations there are two possibilities:
$E = \{\alpha = \beta, \ldots\}$, which can be removed with A6
$E = \{\alpha = \beta^F, \ldots\}$, since $\langle t | \alpha = \Box \ldots \rangle$ is a context and $\beta^F = s^F$ for a guarded s. We have $t^E = \langle t | \alpha = s^F, \ldots \rangle$

Proposition 2.5.5 *Each graph-like term is provably equal to a flat graph-like term.*

Proof Given a graph-like term M one may assume

$$M = \langle \alpha_1 | \vec{\alpha} = \vec{t}(\vec{\alpha}) \rangle = \alpha_1^E$$

with all $t_i(\vec{\alpha})$ guarded. Let $\vec{s}$ denote the list of all variables and all subterms of the $t_i(\vec{\alpha})$'s, with $s_1 = \alpha_1$. We have:

$$s_i^E = F_i(\vec{s})^E,$$

where the F_i's are not necessarily distinct function symbols, because s_i is either guarded or a variable defined as a guarded term in E.
Folding gives that:

$$M = \alpha_1^E = s_1^E = \langle \beta_1 | \vec{\beta} = \vec{F}(\vec{\beta}) \rangle = N$$

where N is a flat term.

□

Proposition 2.5.6 *Each term is provably equal to some graph-like term.*

Proof Given $M = \alpha_1^E$ with

$$E = \{\vec{\alpha} = \vec{t}(\vec{\alpha}, \beta_{1,*}^{\vec{F}_*})\}$$

we can derive the following equations:

$$\alpha_i^E = t_i(\vec{\alpha}^E, {\beta_{1,*}^{\vec{F}_*}}^E)$$

and, assuming $F_i = \{\vec{\beta_{i,*}} = \vec{s_{i,*}}(\vec{\alpha}, \vec{\beta_{i,*}}^{F_i})\}$:

$$\beta_{i,j}^{F_i\,E} = s_{i,j}(\vec{\alpha}^E, \vec{\beta_{i,*}}^{F_i\,E})$$

folding yields:

$$M = \langle \gamma_1 | \vec{\gamma} = \vec{t}(\vec{\gamma}, \vec{\vec{\gamma}}), \vec{\vec{\gamma}} = \vec{\vec{s}}(\vec{\gamma}, \vec{\vec{\gamma}})\rangle$$

□

Example 2.5.1

$$M = \langle \alpha | \underbrace{\alpha = F(\langle \beta | \overbrace{\beta = G(\alpha)}^{F}\rangle)}_{E}\rangle$$

So:

$$\alpha^E = F(\beta^{F^E})$$
$$\beta^{F^E} = G(\alpha^E)$$

Which folds to:

$$M = \alpha^E = \langle \alpha | \alpha = F(\beta), \beta = G(\alpha)\rangle$$

Proposition 2.5.7 *If $\vdash s = t$ is valid for flat and graph-like s and t then the equality is deducible.*

Proof We know that $s = \alpha^E$ and $t = \beta^F$ for programs E and F. The term graphs $E@\alpha$ and $F@\beta$ are bisimilar. Take a bisimulation R. W.l.o.g. $def(E) \cap def(F) = \emptyset$. Put $\Psi = \{\gamma_1 = \gamma_2\}_{(\gamma_1,\gamma_2)\in R}$ and observe that:

$$\vdash \Psi\{\perp/\gamma\}_{\gamma\in def(E)\cup def(F)}$$

and

$$\Psi \vdash \Psi\{t/\gamma\}_{\gamma=t\in E\cup F}$$

are both deducible (in the same way as for term graphs). So by induction:

$$\vdash \Psi^{E\cup F}$$

From this assertion we can deduce:

$$\vdash \alpha^{E\cup F} = \beta^{E\cup F}$$

Applying garbage collection yields:

$$\vdash \alpha^E = \beta^F$$

Theorem 2.5.8 (completeness of proof system VII) *If $\vdash \Phi$ is valid then it is provable.*

Proof Assume $\Phi = \{s_1 = t_1, \ldots, s_n = t_n\}$. We also know that there are flat graph-like terms s'_i and t'_i for $i = 1, \ldots, n$ for which it is deducible that:

$$\vdash \{s_i = s'_i, t_i = t'_i\}_{i=1}^{n}$$

This means that $\vdash \{s'_i = t'_i\}_{i=1}^{n}$ is valid and therefore deducible. Simple equational reasoning deduces from that assertion that

$$\vdash \{s_1 = t_1, \ldots, s_n = t_n\}$$

□

2.5.4 Extending the NTG proof system

We already have completeness for assertions of the form

$$\vdash \Psi$$

To obtain completeness for assertions of the form

$$\Phi \vdash \Psi$$

we need to extend the proof system with at least:

Proof system VIII.

(A14)	$f(\vec{s}) = g(\vec{t}) \vdash S = T$	*falsum*
(A15)	$f(\vec{s}) = f(\vec{t}) \vdash \vec{s} = \vec{t}$	*decomposition*

Theorem 2.5.9 (soundness of proof system VIII) *Every provable assertion is valid.*

Proof

(A14)
The left-hand side can never become true, so the implication will always be true.

(A15)
For trees the following holds:

$$f([\![\vec{s}^E]\!]) = f([\![\vec{t}^E]\!]) \Leftrightarrow [\![\vec{s}^E]\!] = [\![\vec{t}^E]\!]$$

See Proposition 1.1.3.

□

Conjecture 2.5.10 (completeness of proof system VIII) *If*

$$\Phi \vdash \Psi$$

is valid then there is a deduction for it in the extended NTG proof system.

The idea is that all the information in Φ is the restriction made on free variables and wether or not there is a substitution σ that makes $[\![\Phi]\!]\sigma$ true. If such a substitution does not exist then we can use axiom A14 to prove the assertion, otherwise the information about free variables can be derived and merged into Ψ yielding a Ψ' that is valid for all substitutions and therefore provable, so $\Phi \vdash \Psi'$ is provable. From there one can put back the information on the free variables with A13 to prove $\Phi \vdash \Psi$.

Acknowledgements

I am grateful to professor J.W. de Bakker for his help in locating the error in the Courcelle Vuillemin paper and for a quick look he had at the μ-like system to check if I did not make any obvious mistakes.

I am also grateful to dr. Z.M. Ariola for several suggestions concerning this research.

I thank my mother for taking care of me during the writing of this thesis and the preceeding four years of my study and also for the support I got from her and the rest of the family.

Finally I would like to thank my supervisor Jan Willem Klop for his time and for making this report look considerably more human than it would have been without his influence.

Appendix: The proof systems

μ-calculus like

Proof system IX.

(A1)	$\mathcal{P} \vdash t \leq t$	*reflexivity*
(A2)	$\mathcal{P}, s \leq t, t \leq u \vdash s \leq u$	*transitivity*
(A3)	$\mathcal{P}, s \leq t, t \leq s \vdash s = t$	*anti-symmetry*
(A4)	$\mathcal{P}, s = t \vdash s \leq t, t \leq s$	*anti-symmetry*
(A5)	$\mathcal{P} \vdash t = t$	*reflexivity*
(A6)	$\mathcal{P}, s = t \vdash t = s$	*symmetry*
(A7)	$\mathcal{P}, s = t, t = u \vdash s = u$	*transitivity*
(A8)	$\mathcal{P}, \Phi \vdash \Phi$	*quotation*
(A9)	$\mathcal{P} \vdash \perp \leq t$	*minimality*
(A10)	$\mathcal{P}, \{s_1 \leq t_1, \ldots, s_n \leq t_n\} \vdash f(s_1, \ldots, s_n) \leq f(t_1, \ldots, t_n)$	*monotonicity*
(A11)	$\mathcal{P} \vdash t = t\{s/\alpha\}$, if $\alpha = s \in \mathcal{P}$	*program*
(R1)	$\dfrac{\mathcal{P}, \Phi \vdash \Psi}{\mathcal{P}, \Phi' \vdash \Psi'}$, for $\Phi \subset \Phi'$ and $\Psi \supset \Psi'$.	*inclusion*
(R2)	$\dfrac{\mathcal{P}, \Phi_1 \vdash \Phi_2 \quad \mathcal{P}, \Phi_2 \vdash \Phi_3}{\mathcal{P}, \Phi_1 \vdash \Phi_3}$	*cut*
(R3)	$\dfrac{\mathcal{P}, \Phi_1 \vdash \Psi_1 \quad \mathcal{P}, \Phi_2 \vdash \Psi_2}{\mathcal{P}, \Phi_1 \cup \Phi_2 \vdash \Psi_1 \cup \Psi_2}$	*conjunction*
(R4)	$\dfrac{\mathcal{P}, \Phi \vdash \Psi}{\mathcal{P}', \Phi \vdash \Psi}$, if $\mathcal{P} \subset \mathcal{P}'$.	*inclusion*
(R5)	$\dfrac{\mathcal{P} \cup \{\alpha = C[\beta], \beta = t\}, \Phi \vdash \Psi}{\mathcal{P} \cup \{\alpha = C[t], \beta = t\}, \Phi \vdash \Psi}$	*substitution*
(R6)	$\dfrac{\mathcal{P}, \Phi \vdash \Psi}{\mathcal{P}', \Phi \vdash \Psi}$, if $\mathcal{P} \supset \mathcal{P}'$ and no new free variables appear.	*garbage*
(R7)	$\dfrac{\mathcal{P}, \Phi \vdash \Psi\{\perp/\alpha_i\}_{i=1}^{n} \quad \mathcal{P}, \Phi, \Psi \vdash \Psi\{t_i/\alpha_i\}_{i=1}^{n}}{\mathcal{P} \cup \{\alpha_1 = t_1, \ldots, \alpha_n = t_n\}, \Phi \vdash \Psi}$, if no α_i occurs in $\mathcal{P}$ or Φ and no t_i is a variable.	*induction*

This proof system is the full version of proof system V.

μ-equality

Proof system X.

(A1)	$\mathcal{P} \vdash t = t$	*reflexivity*
(A2)	$\mathcal{P}, s = t \vdash t = s$	*symmetry*
(A3)	$\mathcal{P}, s = t, t = u \vdash s = u$	*transitivity*
(A4)	$\mathcal{P}, \Phi \vdash \Phi$	*quotation*
(A5)	$\mathcal{P}, \{s_1 = t_1, \ldots, s_n = t_n\} \vdash f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)$	*congruence*
(A6)	$\mathcal{P} \vdash t = t\{s/\alpha\}$, if $\alpha = s \in \mathcal{P}$	*program*
(R1)	$\dfrac{\mathcal{P}, \Phi \vdash \Psi}{\mathcal{P}, \Phi' \vdash \Psi'}$, for $\Phi \subset \Phi'$ and $\Psi \supset \Psi'$.	*inclusion*
(R2)	$\dfrac{\mathcal{P}, \Phi_1 \vdash \Phi_2 \quad \mathcal{P}, \Phi_2 \vdash \Phi_3}{\mathcal{P}, \Phi_1 \vdash \Phi_3}$	*cut*
(R3)	$\dfrac{\mathcal{P}, \Phi_1 \vdash \Psi_1 \quad \mathcal{P}, \Phi_2 \vdash \Psi_2}{\mathcal{P}, \Phi_1 \cup \Phi_2 \vdash \Psi_1 \cup \Psi_2}$	*conjunction*
(R4)	$\dfrac{\mathcal{P}, \Phi \vdash \Psi}{\mathcal{P}', \Phi \vdash \Psi}$, if $\mathcal{P} \subset \mathcal{P}'$.	*inclusion*
(R5)	$\dfrac{\mathcal{P} \cup \{\alpha = C[\beta], \beta = t\}, \Phi \vdash \Psi}{\mathcal{P} \cup \{\alpha = C[t], \beta = t\}, \Phi \vdash \Psi}$	*substitution*
(R6)	$\dfrac{\mathcal{P}, \Phi \vdash \Psi}{\mathcal{P}', \Phi \vdash \Psi}$, if $\mathcal{P} \supset \mathcal{P}'$ and no new free variables appear.	*garbage*
(R7)	$\dfrac{\mathcal{P}, \Phi \vdash \Psi\{\perp/\alpha_i\}_{i=1}^n \quad \mathcal{P}, \Phi, \Psi \vdash \Psi\{t_i/\alpha_i\}_{i=1}^n}{\mathcal{P} \cup \{\alpha_1 = t_1, \ldots, \alpha_n = t_n\}, \Phi \vdash \Psi}$, if no α_i occurs in $\mathcal{P}$ or Φ and no t_i is a variable.	*induction*

This proof system is the full version of proof system VI.

Nested Term Graphs

Proof system XI.

(A1)	$\vdash t = t$	*reflexivity*
(A2)	$s = t \vdash t = s$	*symmetry*
(A3)	$s = t, t = u \vdash s = u$	*transitivity*
(A4)	$\Phi \vdash \Phi$	*quotation*
(A5)	$\vdash \langle t \mid \alpha = \alpha, E\rangle = \langle t \mid \alpha = \bullet, E\rangle$	*black hole*
(A6)	$\vdash \langle t \mid \alpha = \beta, E\rangle = \langle t \mid E\rangle\{\beta/\alpha\}$, where α and β are different variables	*renaming*
(A7)	$\vdash \langle \alpha \mid \alpha = t, E\rangle = \langle t \mid \alpha = t, E\rangle$	*substitution*
(A8)	$\vdash s = t$, if $s \rightarrow_\alpha t$	*alpha conversion*
(A9)	$\langle s \mid E\rangle = \langle s \mid F\rangle$, if $E = F$ modulo garbage collection	*garbage collection*
(A10)	$\vdash \langle f(s_1, \ldots, s_n) \mid E\rangle = f(\langle s_1 \mid E\rangle, \ldots, \langle s_n \mid E\rangle)$	*unwinding*
(A11)	$\vdash \langle t \mid \rangle = t$	*box-removal*
(A12)	$\vdash \langle \langle s \mid E\rangle \mid F\rangle = \langle s \mid E, F\rangle$, if $Def(E) \cap Var(F) = \emptyset$.	*merging*
(A13)	$s = t \vdash C[s] = C[t]$, for any context $C[.]$ that does not capture free variables from s or t.	*context*
(R1)	$\dfrac{\Phi \vdash \Psi}{\Phi' \vdash \Psi'}$, for $\Phi \subset \Phi'$ and $\Psi \supset \Psi'$.	*inclusion*
(R2)	$\dfrac{\Phi_1 \vdash \Phi_2 \quad \Phi_2 \vdash \Phi_3}{\Phi_1 \vdash \Phi_3}$	*cut*
(R3)	$\dfrac{\Phi_1 \vdash \Psi_1 \quad \Phi_2 \vdash \Psi_2}{\Phi_1 \cup \Phi_2 \vdash \Psi_1 \cup \Psi_2}$	*conjunction*
(R4)	$\dfrac{\vdash s' = t'}{\vec{s} = \vec{t} \vdash s'\{\vec{s}/\vec{\alpha}\} = t'\{\vec{t}/\vec{\alpha}\}}$	*substitution*
(R5)	$\dfrac{\vdash s = t}{\vdash C[s] = C[t]}$, for any context $C[.]$	*context*
(R6)	$[s_i = t_i(\vec{s})]_{i=1}^n \vdash s_1 = \langle \alpha_1 \mid [\alpha_i = t_i(\vec{\alpha})]_{i=1}^n\rangle$, if each $t_i(\vec{\beta})$ is guarded and all α_i are fresh.	*folding*
(R7)	$\dfrac{\Phi \vdash \Psi\{s_i/\alpha_i\}_{i=1}^n \quad \Phi, \Psi \vdash \Psi\{t_i/\alpha_i\}_{i=1}^n}{\Phi \vdash \Psi^{\{\alpha_i = t_i\}_{i=1}^n}}$, for any s_i, if no α_i is free in Φ and each t_i is guarded.	*induction*

This proof system is the full version of proof system VII.

Equational Logic.

Proof system XII: EL

(A1)	$\vdash t = t$		*reflexivity*
(R1)	$\dfrac{\vdash t_1 = t_2}{\vdash t_2 = t_1}$		*symmetry*
(R2)	$\dfrac{\vdash t_1 = t_2, t_2 = t_3}{\vdash t_1 = t_3}$		*transitivity*
(R3)	$\dfrac{\vdash t = s}{\vdash t\sigma = s\sigma}$, for any substitution σ		*substitution*
(R4)	$\dfrac{\vdash \vec{t} = \vec{s}}{\vdash f(\vec{t}) = f(\vec{s})}$		*congruence*

In this notation Equational Logic is a subset of the proof system for nested term graphs.

Equational Logic extended to μ-terms

Proof system XIII: EL_μ

(A1)	$\vdash t = t$	*reflexivity*
(R1)	$\dfrac{\vdash t_1 = t_2}{\vdash t_2 = t_1}$	*symmetry*
(R2)	$\dfrac{\vdash t_1 = t_2, t_2 = t_3}{\vdash t_1 = t_3}$	*transitivity*
(R3)	$\dfrac{\vdash t = s, t' = s'}{\vdash t\{t'/x\} = s\{s'/x\}}$, for a variable x	*substitution*
(A2)	$\vdash \mu\alpha.t(\alpha) = t(\mu\alpha.t(\alpha))$	*unwinding*
(A3)	$\vdash \mu\alpha.t(\alpha) = \mu\beta.t(\beta)$, if β does not occur free in $t(\alpha)$	*renaming*
(R4)	$\dfrac{\vdash t_1 = t(t_1)}{\vdash t_1 = \mu\alpha.t(\alpha)}$, $t(\alpha)$ guarded and α does not occur free in $t(.)$	*folding*

If we translate the μ construct to a nested term graph:

$$\mu\alpha.t(\alpha) = \langle\alpha|\alpha = t(\alpha)\rangle$$

the folding rule in EL_μ is a special case of the NTG folding axiom and EL_μ becomes a subsystem of the NTG proof system.

References

[1] Z.M. Ariola. Relating Graph and Term Rewriting via Böhm Models. In C. Kirchner, editor, *Proceedings RTA '93*, pages 183–197. Montreal, Springer-Verlag LNCS 690, 1993.

[2] Z.M. Ariola and J.W. Klop. Equational Term Graph Rewriting. CWI-report CS-R94xx.

[3] B. Courcelle, G. Kahn, and J. Vuillemin. Algorithmes d'équivalence et de réduction à des expressions minimales dans une classe d'équations récursive simples. In *Proc.* 2$^{\text{nd}}$ *Colloquium on Automata, Languages and Programming*, pages 200–213. Springer-Verlag LNCS 407, 1974.

[4] B. Courcelle and J. Vuillemin. Completeness results for the equivalence of recursive program schemes. *JCCS*, 12:179–197, 1976.

[5] J.W. de Bakker. *Recursive procedures*, volume 24 of *Mathematical Centre Tracts.* Mathematisch Centrum Amsterdam, 1971.

[6] N. Dershowitz and J.P. Jouannaud. Rewrite systems. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 243–320. Elsevier Science Publishers BV, Amsterdam, 1990.

[7] J.W. Klop. Term Rewriting Systems. In S. Abramsky, Dov M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Theoretical Computer Science*, volume 2, pages 1–116. Oxford University Press, New York, 1992.

[8] S. Smetsers. *Graph Rewriting and Functional Languages.* PhD thesis, University of Nijmegen, 1993.

List of symbols

Index