Nested Term Graphs

Clemens Grabmayer
Department of Computer Science
VU University Amsterdam
The Netherlands
C.A.Grabmayer@vu.nl

Vincent van Oostrom
Philosophy
Utrecht University
The Netherlands
V.vanOostrom@uu.nl

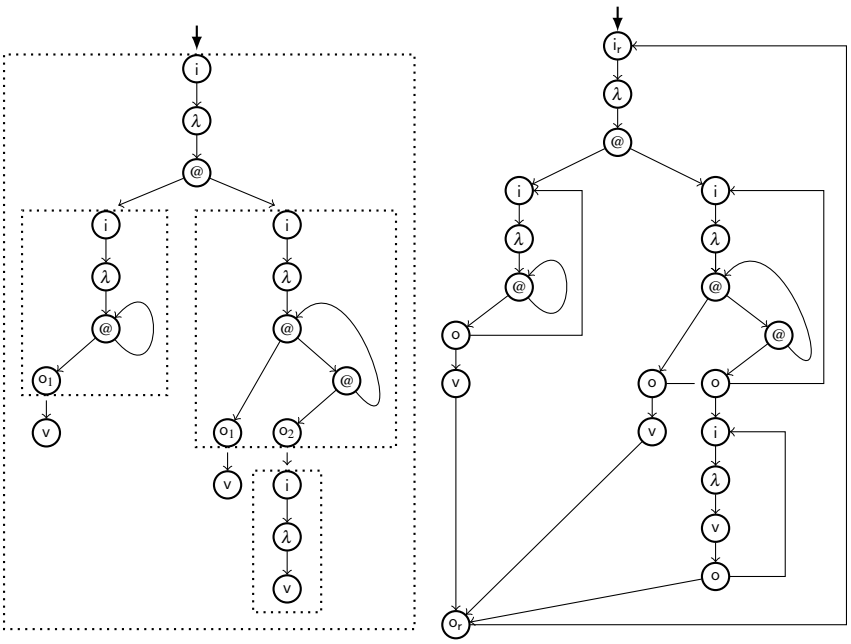
We report on work in progress on 'nested term graphs' for formalizing higher-order terms (e.g. finite or infinite λ-terms), including those expressing recursion (e.g. terms in the λ-calculus with letrec). The idea is to represent the nested scope structure of a higher-order term by a nested structure of term graphs. Based on a signature that is partitioned into atomic and nested function symbols, we define nested term graphs both intensionally, as tree-like recursive graph specifications that associate nested symbols with usual term graphs, and extensionally, as enriched term graph structures. These definitions induce corresponding notions of bisimulation between nested term graphs. Our main result states that nested term graphs can be implemented faithfully by first-order term graphs.

Introduction Structures such as strings, terms, and graphs frequently come equipped with additional structure. In this paper we study the case where this extra structure is a notion of scope, for the particular case of term graphs. Scopes are abundant in programming and in logic. The guiding intuition is that the notion of scope corresponds to a notion of context-freeness. We illustrate this first by means of a string example, which although very simple already illustrates the issues involved. Consider the ad hoc context free grammar for expressions $S ::= 2 \times T$, $T ::= 3 + 1$. The obvious intended interpretation into the natural numbers of T is 4 and that of S is then $2 \times 4 = 8$. The standard observation is that interpretation does not commute with transformations of the grammar. In particular, substituting $3 + 1$ for T in the definition of S yields $S ::= 2 \times 3 + 1$ for which one obtains $6 + 1 = 7$ as interpretation, which is not exactly the same as 8. A way to prevent such a misinterpretation is to insert parentheses first to indicate the right scope: $S ::= 2 \times (3 + 1)$. We turn this observation around by stipulating that the notion of scope is a phenomenon that is brought about by context-free recursive specifications, for strings, terms, and graphs alike. An example for terms in programming (Lisp) is unhygienic macro expansion [5]: $(\mathsf{or}\,\langle exp\rangle_1\langle exp\rangle_2) ::= (\mathsf{let}\,v[\,]_{\langle exp\rangle_1}(\mathsf{if}\,v\,v[\,]_{\langle exp\rangle_2}))$. Expanding this macro in $(\mathsf{or}\,\mathsf{nil}\,v)$ yields $(\mathsf{let}\,v\,\mathsf{nil}\,(\mathsf{if}\,v\,v\,v))$ which always yields nil due to the inadvertent capturing of v. In this case, a way to prevent such a misinterpretation is to insert λs, a device from [4] for ending scopes of binders, resulting in $(\mathsf{let}\,v\,\lambda v.\mathsf{nil}\,(\mathsf{if}\,v\,v\,\lambda v.v))$ avoiding that the substituted v becomes bound by the $\mathsf{let}\,v$ of the macro by unbinding the latter by λv. Here we are concerned with the same phenomenon but for term graphs and their behavorial semantics. As a running example we use the following expression, which expresses a cyclic λ-term (and thereby a regular infinite λ-term) by means of the Combinatory Reduction System (CRS) inspired gletrec-notation:

$$
\begin{array}{lrcl}
\mathsf{gletrec} & \mathsf{n}() & ::= & \lambda x.\mathsf{f}_1(x)\mathsf{f}_2(x,\mathsf{g}()) \\
 & \mathsf{f}_1(X_1) & ::= & \lambda x.\mathsf{let}\,\alpha = X_1\alpha\,\mathsf{in}\,\alpha \\
 & \mathsf{f}_2(X_1,X_2) & ::= & \lambda y.\mathsf{let}\,\beta = X_1(X_2\beta)\,\mathsf{in}\,\beta \\
 & \mathsf{g}() & ::= & \lambda z.z \\
\mathsf{in} & \mathsf{n}() & &
\end{array}
$$

Submitted to:
TERMGRAPH 2014

which corresponds to the pretty printed 'recursive graph specification' on the left (the graph with scopes indicated by dotted lines). Our main result entails that the behavorial semantics of this specification is the same as that of the first-order term graph obtained from it, displayed on the right here. Note that in this first-order term graph additional vertices and edges between them have been inserted to delimit scopes appropriately; they play the same rôle as the parentheses in the string example and the λ in the term example. This example belongs to a particularly well-behaved subclass of the recursive graph specifications, the so-called nested term graphs, where the dependency between the nested symbols (n, f_1, f_2, g in the example) is tree-like. In particular, the first-order term graph can be interpreted as a λ-term graph [3] or a higher-order term graph [1]. However, our results pertain to specifications with arbitrary dependencies, allowing for both sharing and cyclicity, such as:



$$\begin{array}{lrcl} \mathsf{gletrec} & \mathsf{f}() & ::= & \lambda x.\mathsf{g}(x) \\ & \mathsf{g}(X_1) & ::= & \lambda y.\mathsf{g}(y)X_1 \\ \mathsf{in} & \mathsf{f}() & & \end{array}$$

Ordinary term graphs. Let Σ be a (first-order) signature for function symbols with arity function $ar : \Sigma \to \mathbb{N}$. A *term graph over* Σ (a Σ*-term-graph*) is a tuple $\langle V, lab, args, root\rangle$ where V is a set of *vertices*, $lab : V \to \Sigma$ the *(vertex) label function*, $args : V \to V^*$ the *argument function* that maps every vertex v to the word $args(v)$ consisting of the $ar(lab(v))$ successor vertices of v (hence it holds $|args(v)| = ar(lab(v))$), and $root \in V$ is the *root* of the term graph. Such a term graph is called *root-connected* if every vertex is reachable from the root by a path that arises by repeatedly going from a vertex to one of its successors. By $\mathrm{TG}(\Sigma)$ we denote the class of all root-connected term graphs over Σ. *Note:* By a 'term graph' we will mean by default a root-connected term graph.

A *rooted* ARS is the extension of an abstract rewriting system (ARS) $\to$ by specifying one of its objects as designated *root*. A rooted ARS $\to$ with objects A and root a is called a *tree* if $\to$ is acyclic (there is no $x \in A$ such that $x \to^+ x$), co-deterministic (for every $x \in A$ there is at most one step of $\to$ with target x), and root-connected (every element $x \in A$ is reachable from a via a sequence of steps of $\to$, i.e. $a \to^* x$).

Nested Term Graphs A *signature for nested term graphs* (an *ntg-signature*) is a signature Σ for term graphs that is partitioned into a part Σ_{at} for *atomic* symbols, and a part Σ_{ne} for *nested* symbols, that is, $\Sigma = \Sigma_{\mathrm{at}} \cup \Sigma_{\mathrm{ne}}$ and $\Sigma_{\mathrm{at}} \cap \Sigma_{\mathrm{ne}} = \varnothing$. In addition to a given signature Σ for nested term graphs we always assume additional *interface symbols* from the set $IO = I \cup O$, where $I = \{\mathsf{i}\}$ consists of a single unary *input* symbol (symbolizing an input edge into a term graph), and $O = \{\mathsf{o}_1, \mathsf{o}_2, \mathsf{o}_3, \ldots\}$ a countably infinite set of *output* symbols with arity zero (symbolizing a numbered output edge from a term graph).

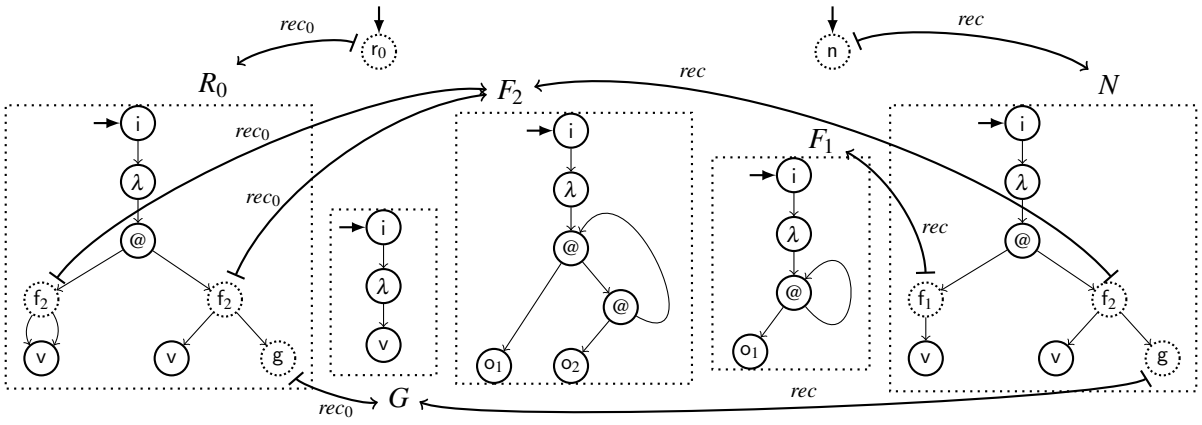


Figure 1: Definitions of a recursive graph specification $\mathcal{R}_0$ (Ex. 2), and a nested term graph $\mathcal{N}$ (Ex. 4).

Definition 1 (recursive specifications for nested term graphs)**.** Let Σ be a signature for nested term graphs. A *recursive (nested term) graph specification* (an *rgs)* over Σ is a tuple $\langle rec, r\rangle$, where:

- $rec : \Sigma_{\mathrm{ne}} \to \mathrm{TG}(\Sigma \cup IO)$ is the *specification function* that maps a nested function symbol $f \in \Sigma_{\mathrm{ne}}$ with $ar(f) = k$ to a term graph $rec(f) = F \in \mathrm{TG}(\Sigma \cup \{\mathsf{i}, \mathsf{o}_1, \ldots, \mathsf{o}_k\})$ that has precisely one vertex labeled by i, the root, and that contains precisely one vertex labeled by the o_i, for each $i \in \{1, \ldots, k\}$;
- $r \in \Sigma_{\mathrm{ne}}$, a nullary symbol (that is, $ar(r) = 0$), is the *root symbol*.

For such an rgs $\mathcal{R} = \langle rec, r\rangle$ over Σ, the rooted *dependency* ARS $\circ\!\!-$ of $\mathcal{R}$ has as objects the nested symbols in Σ_{ne}, it has root r, and the following steps: for all $f, g \in \Sigma_{\mathrm{ne}}$ such that g occurs in the term graph $rec(f)$ at position p there is a step $p : f \circ\!\!- g$.

Example 2. Let $\Sigma_{\mathrm{at}} = \{\lambda/1, @/2, \mathsf{v}/0\}$ for expressing λ-terms as term graphs.

(i) Let $\Sigma_{0,\mathrm{ne}} = \{\mathsf{r}_0/0, \mathsf{f}_2/2, \mathsf{g}/0\}$. Then $\mathcal{R}_0 = \langle rec_0, \mathsf{r}_0\rangle$, where $rec_0 : \Sigma_{0,\mathrm{ne}} \to \mathrm{TG}(\Sigma \cup IO)$ is defined by $\mathsf{r}_0 \mapsto R_0$, $\mathsf{f}_2 \mapsto F_2$, and $\mathsf{g} \mapsto G$ as shown in Fig. 1, is an rgs.

(ii) Let $\Sigma_{\mathrm{ne}} = \{\mathsf{n}/0, \mathsf{f}_1/1, \mathsf{f}_2/2, \mathsf{g}/0\}$. Then $\langle rec, \mathsf{n}\rangle$, where $rec : \Sigma_{\mathrm{ne}} \to \mathrm{TG}(\Sigma \cup IO)$ is defined by $\mathsf{n} \mapsto N$, $\mathsf{f}_1 \mapsto F_1$, $\mathsf{f}_2 \mapsto F_2$, and $\mathsf{g} \mapsto G$ as shown in Fig. 1, is an rgs.

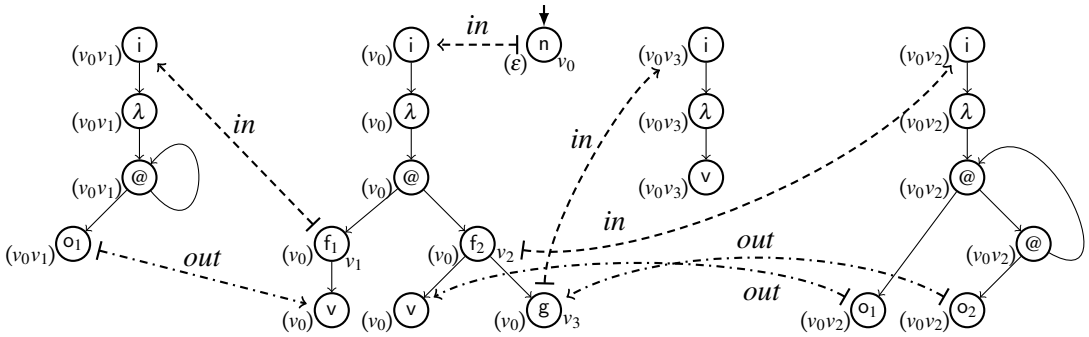
Definition 3 (nested term graphs)**.** Let Σ be an ntg-signature. A *nested term graph* (an *ntg*) over Σ is an rgs $\mathcal{N} = \langle rec, r\rangle$ such that the rooted dependency ARS $\circ\!\!-$ is a tree. By $\mathrm{NG}(\Sigma)$ we denote the class of nested term graphs over Σ.

Example 4. We first consider the rgs $\mathcal{R}_0 = \langle rec_0, \mathsf{r}_0\rangle$ from Ex. 2, (i). Its rooted dependency ARS $\circ\!\!-$ is not a tree, because there are two steps that witness $\mathsf{r}_0 \circ\!\!- \mathsf{f}_2$, namely those that are induced by the two occurrences of f_2 in the term graph $R_0 = rec_0(\mathsf{r}_0)$. As a consequence, $\mathcal{R}_0$ is not a nested term graph.

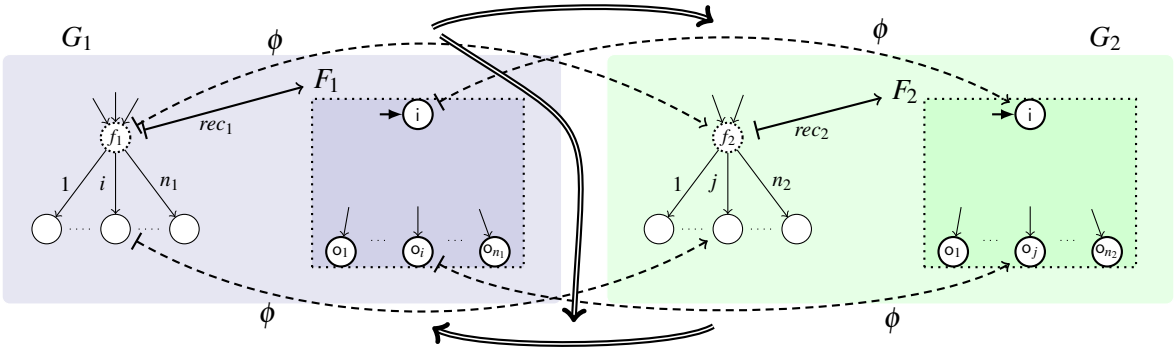
But for the rgs $\mathcal{N} = \langle rec, \mathsf{n}\rangle$ from Ex. 2, (ii), we find that the rooted dependency ARS $\circ\!\!-$ actually is a tree with root n. Hence $\mathcal{N}$ is a nested term graph. For a 'pretty print' of $\mathcal{N}$, see the left graph on page 2.

There is an easy correspondence between nested term graphs defined by the intensional definition above, and ntgs according to an extensional definition as enrichments of ordinary term graphs. An *extensional description* of a nested term graph (an *entg's*) is a tuple $\langle V, lab, args, in, out, anc, root\rangle$, where $G_0 = \langle V, lab, args, root\rangle$ is a (not necessarily root-connected) term graph over $\Sigma \cup IO$, $in : V \rightharpoonup V$ is a partial function that maps a vertex v labeled by a nested symbol to the root of the term graph nested into v, $out : V \rightharpoonup V$ is a partial function that to every output vertex o_i assigns the i-th successor of the vertex into which the term graph containing o_i is nested, and $anc : V \to V^*$ is the *ancestor function* that records, and

guarantees, the nesting structure by assigning to every vertex v the word $anc(v) = v_1 \cdots v_n$ made up of the vertices in which v is nested. As an example consider the illustration of an entg representation of the ntg $\mathcal{N}$ in Ex. 4, with names for vertices with nested symbols (right of such vertices), and the values of the ancestor function indicated in brackets (left of the vertices):



Nested Bisimulation Let $\mathcal{R}_1 = \langle rec_1, r_1 \rangle$ and $\mathcal{R}_2 = \langle rec_2, r_2 \rangle$ be rgs's over signatures Σ_1 and Σ_2 with the same atomic symbols. A *homomorphism* between rgs's $\mathcal{R}_1$ and $\mathcal{R}_2$ (denoted by $\mathcal{R}_1 \rightarrow\!\!\!\!\rightarrow \mathcal{R}_2$) is a function $\phi : V_1 \to V_2$ between the set of vertices of the disjoint unions G_1 and G_2 of the term graphs in the image of rec_1 and rec_2, respectively; on the vertices of G_1 and G_2 labeled with atomic, or interface labels, ϕ behaves like an ordinary term graph homomorphism; and on the vertices labeled with nested symbol f_1 and f_2, the following 'interface' clause applies:



A *bisimulation* between $\mathcal{R}_1$ and $\mathcal{R}_2$ (denoted by $\mathcal{R}_1 \leftrightarrows \mathcal{R}_2$) is an rgs $\mathcal{R}$ such that $\mathcal{R}_1 \leftarrow\!\!\!\!\leftarrow \mathcal{R} \rightarrow\!\!\!\!\rightarrow \mathcal{R}_2$.

A *nested bisimulation* between rgs's $\mathcal{R}_1$ and $\mathcal{R}_2$ compares rgs's in a finer manner by recording also the nesting behaviour of the rgs's by means of stacks of vertices. It is defined similarly between prefixed expressions $(v_1 \cdots v_{k_1})v$ and $(w_1 \cdots w_{k_2})w$ that describe visits of the vertices v_1 in G_1 and w_1 in G_2 in the context of histories of visits as recorded by the stacks $v_1 \cdots v_{k_1}$ and $w_1 \cdots w_{k_2}$ of nested vertices of G_1 and G_2. The topmost stack element always indicates the parent nested vertex, enabling a definition by local progression clauses. We denote bisimilarity by $\leftrightarrows^{\mathrm{ne}}$ and functional bisimilarity by $\rightarrow\!\!\!\!\rightarrow^{\mathrm{ne}}$.

For an example illustrating these relations on nested term graphs, see Fig. 2. While on rgs's, $\rightarrow\!\!\!\!\rightarrow^{\mathrm{ne}}$ is properly contained in $\rightarrow\!\!\!\!\rightarrow$, these relations coincide on nested term graphs (due to the tree structure of $\multimap$). As a consequence, also $\leftrightarrows^{\mathrm{ne}}$ and $\leftrightarrows$ coincide on ntgs. If $\mathcal{R}_1 \leftrightarrows^{\mathrm{ne}} \mathcal{R}_2$ holds for rgs's $\mathcal{R}_1$ and $\mathcal{R}_2$, then the term graphs $\mathcal{N}(\mathcal{R}_1)$ and $\mathcal{N}(\mathcal{R}_2)$ specified by $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively, are isomorphic.

Implementation by first-order term graphs Nested term graphs can be implemented in a faithful, and rather natural way as first-order term graphs. By 'faithful' we mean that the interpretation mapping is a retraction that preserves and reflects homomorphisms, and by 'natural' that it can be defined inductively on

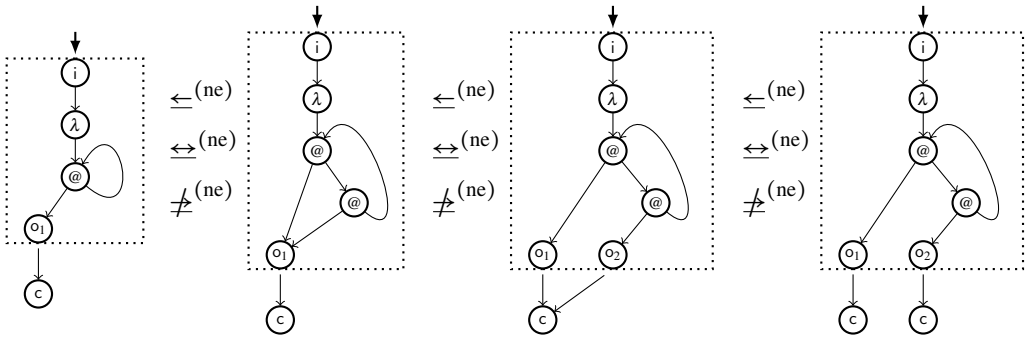


Figure 2: Four simple nested term graphs that are related by $\twoheadrightarrow$ and $\twoheadrightarrow^{\mathrm{ne}}$, and by $\leftrightarrow\!\!\!\!\!\twoheadrightarrow$ and $\leftrightarrow\!\!\!\!\!\twoheadrightarrow^{\mathrm{ne}}$.

the nesting structure. The basic idea is analogous to the interpretation of λ-ho-term-graphs as first-order λ-term-graphs in [3]. For a nested term graph $\mathcal{N} = \langle rec, r\rangle$ its first-order term graph interpretation $T(\mathcal{N})$ is defined over $\Sigma' = \Sigma \cup I \cup \{\mathsf{o}/2, \mathsf{i_r}/1, \mathsf{o_r}/1\}$ by repeatedly replacing, starting on the term graph $rec(r)$, a vertex v with a nested symbol f by the term graph specification $rec(f)$ of f, thereby directing incoming edges at v to the root v_r of $rec(f)$, and replacing output vertices o_i of $rec(f)$ by the binary symbol o with the first edge targeting the i-th successor of v, and the second edge being a back-link to v_r. Beforehand, the ntg has been pre-processed by replacing vertices with atomic constants by vertices with fresh unary labels, and with edges to a (per nested symbol) single additional output vertex. Input/output vertices at the root level get label $\mathsf{i_r}/\mathsf{o_r}$. For an example, see page 2 for the interpretation $T(\mathcal{N})$ of the ntg $\mathcal{N}$ in Ex. 4.

Theorem 5 (implementation of ntgs by first-order term graphs)**.** *Let Σ be an ntg-signature, and $\Sigma' = \Sigma \cup I \cup \{\mathsf{o}/2, \mathsf{i_r}/1, \mathsf{o_r}/1\}$. There are functions $T : \mathrm{NG}(\Sigma) \to \mathrm{TG}(\Sigma')$ and $\mathcal{N} : \mathrm{TG}(\Sigma') \to \mathrm{NG}(\Sigma)$ between the classes of ntgs over Σ and term graphs over Σ' such that $\mathcal{N} \circ T = \mathrm{id}_{\mathrm{NG}(\Sigma)}$, (i.e. T is a retraction of $\mathcal{N}$, and $\mathcal{N}$ is a section of T) that are efficiently computable, and preserve and reflect functional bisimilarity $\twoheadrightarrow$.*

As a consequence, various well-known results for term graphs can be transferred to nested term graphs. For instance, that every nested term graph $\mathcal{N}$, has, up to isomorphism, a unique nested term graph collapse, and that the bisimulation equivalence class of $\mathcal{N}$ (up to isomorphism) forms a complete lattice w.r.t. $\twoheadrightarrow$.

Further aims We are interested in, and have started to investigate, the following further topics:
Context-free graph grammars. We want to view rgs's as context-free graph grammars in order to recognize rgs-generated nested term graphs as context-free graphs. We expect to find a close connection.
Monadic formulation. We would like to obtain a categorical semantics via algebras and coalgebras: nested term graphs as monads over some signature (cf. [2]), in order to isolate the abstract essence of the nested term graph concept, and, in particular, the implementation of nested term graphs as first-order term graphs.
Rewrite theory for nested term graphs. As higher-order terms have a natural interpretation as nested term graphs, it is desirable to investigate implementations of higher-order rewriting by nested term graph rewriting, and eventually, via the correspondence above, by first-order term graph rewriting.

References

[1] S. Blom (2001): *Term Graph Rewriting – Syntax and Semantics*. Ph.D. thesis, Vrije Universiteit Amsterdam.

[2] N. Ghani, C. Lüth & F. de Marchi (2005): *Monads of coalgebras: rational terms and term graphs*. *Mathematical Structures in Computer Science* 15, pp. 433–451.

[3] C. Grabmayer & J. Rochel (2013): *Term Graph Representations for Cyclic Lambda Terms*. In: *Proceedings of TERMGRAPH 2013*, *EPTCS* 110, pp. 56–73. Extending report: arXiv:1308.1034.

[4] D. Hendriks & V. van Oostrom (2003): $\lambda\!\!\!\!\!\lambda$. In F. Baader, editor: *CADE-19*, *LNAI* 2741, Springer, pp. 136–150.

[5] E. Kohlbecker, D.P. Friedman, M. Felleisen & B. Duba (1986): *Hygienic Macro Expansion*. In: *Proceedings of the 1986 ACM Conference on LISP and Functional Programming*, LFP '86, ACM, pp. 151–161.