Anagopos: A Reduction Graph Visualizer for Term Rewriting and Lambda Calculus

Niels Bjørn Bugge Grathwohl[1], Jeroen Ketema[2], Jens Duelund Pallesen[1], and Jakob Grue Simonsen[1]

1 Department of Computer Science, University of Copenhagen (DIKU)
Njalsgade 126–128, 2300 Copenhagen S, Denmark
{bugge,pallesen,simonsen}@diku.dk

2 Faculty EEMCS, University of Twente
PO Box 217, 7500 AE Enschede, The Netherlands
j.ketema@ewi.utwente.nl

Abstract

We present Anagopos, an open source tool for visualizing reduction graphs of terms in lambda calculus and term rewriting. Anagopos allows step-by-step generation of reduction graphs under six different graph drawing algorithms. We provide ample examples of graphs drawn with the tool.

1998 ACM Subject Classification F.3.1, F.4.2, H.5.0, I.3.0

Keywords and phrases term rewriting, lambda calculus, reduction graphs, visualization

Digital Object Identifier 10.4230/LIPIcs.RTA.2011.61

Category System Description

1 Introduction

Anagopos[1] is a tool for visualizing reduction graphs (see, e.g., Figure 1). We created the software with the following two goals in mind:

Automation Allow for the drawing of large numbers of reduction graphs, which is infeasible by hand; this will hopefully allow researchers to formulate new hypotheses regarding the topological properties of reduction graphs.

Visualization Allow for the dynamics of rewriting to be shown more clearly to students. For example, in the case of the Church-Rosser Theorem, it is often hard for students to comprehend the dynamics when moving beyond either single step reductions, or beyond the tiling diagram usually drawn in the proof for orthogonal systems.

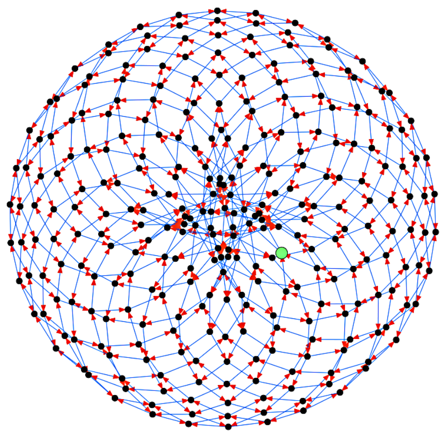


Figure 1 The sphere-like reduction graph of $(ttI)(ttI)$ with $I = \lambda x.x$ and $t = \lambda xy.\vec{y}xxy$, where $\vec{y} = \underbrace{yy\cdots y}_{15}$.

Anagopos can draw reduction graphs of both (untyped) lambda terms and terms from (first-order) term rewriting. To provide immediate

[1] Roughly a contraction of the Greek words anagogí (reduction) and tópos (place).

access to a large number of predefined term rewriting systems, the tool supports the XML-format used by the Termination Problems Data Base (TPDB) from version 7 onwards[2]. As it is a priori not clear what constitutes a good layout for a reduction graph, Anagopos supports multiple algorithms for drawing graphs.

Anagopos is implemented in Python 2.6 [23] and available for download from:

`http://code.google.com/p/anagopos/`

Packages for Ubuntu and other Debian-based systems are provided, as well as a binary for Mac OS X. The source code is available under the GNU General Public License (GPL).

Related software. The only other tool capable of drawing reduction graphs seems to be the *traces* function from PLT Redex [21]; it appears to support only a hierarchical graph drawing algorithm (displaying the initial term on the left and drawing successive reducts further and further to the right). More common are tools that compute reductions: The user specifies the starting term of a reduction and the subsequent redexes contracted are then either selected by the user, or by the program following some pre-defined reduction strategy. Roughly, these tools can be classified according to their presentation of terms, this either in some textual form [27, 37], or in the form of a parse tree [20, 24, 33, 31, 5].

Although not a software tool, also worth mentioning in this context for its graphical qualities is the Alligator Eggs puzzle game that teaches lambda calculus to children by representing lambda abstractions as alligators and variables as their eggs [36].

Outline. The paper is structured as follows: In Section 2 we provide background on reduction graphs. In Sections 3, 4, and 5 we describe, respectively, the interface, graph drawing algorithms, and architecture of Anagopos. In Section 6 we conclude, mentioning some directions in which Anagopos can potentially be extended, and suggesting a number of open problems whose solutions could potentially be found with the help of Anagopos.

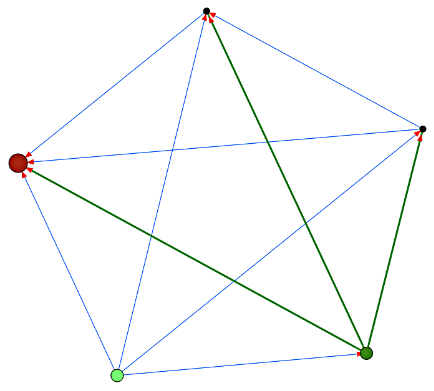
2 Reduction Graphs

Recall that an *abstract rewrite system* (ARS) consists of a set A and a set of binary relations over A; throughout this paper, we consider only a single relation at a time, writing an ARS as a pair $(A, \to)$. Also recall that the abstract reduction system induced by a term rewriting system (TRS) is obtained by letting A be the set of terms and $\to$ be the rewrite relation over the set of terms. The following is a slight adaptation of Definition 1.1.7 in [30]:

▶ **Definition 2.1.** Let $(A, \to)$ be an ARS and let $a \in A$. The *reduction graph* $\mathcal{G}(a)$ of a is the directed graph (V, E) such that V is the set of reducts of a and $(b, c) \in E$ iff $b \to c$.

The literature on reduction graphs for term rewriting systems is sparse, consisting of few general results beyond the standard confluence and Church-Rosser diagrams. For lambda calculus, many basic notions were defined, and conjectures posed, by Barendregt [2] and Klop [16]. The first comprehensive study of reduction graphs appears to be by Venturini Zilli [35, 34]. Subsequent studies by Hirokawa and Sekimoto, and Intrigila and Laurenzi have disproved several early conjectures concerning reduction graphs [26, 11]. Intrigila and Venturini Zilli have investigated representability of ordinals as reduction graphs of lambda terms [12].

[2] See `http://www.termination-portal.org/` for the database and details on the XML-format.


(a) $\mathcal{G}(k_5)$ whose underlying undirected graph is K_5.

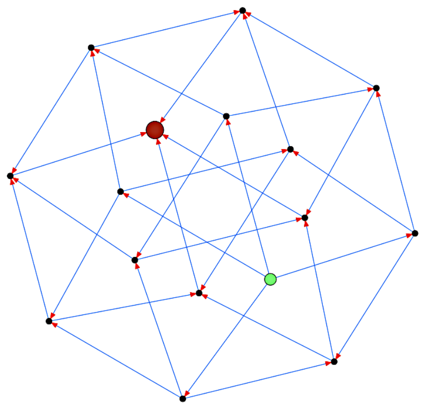

(b) $\mathcal{G}(c_4)$ whose underlying undirected graph is the 4-dimensional hypercube.

Figure 2 The reduction graphs of two of the lambda terms defined in Proposition 2.3. Both graphs are drawn with the Neato drawing algorithm (see Section 4).

Very few general, positive results are known for reduction graphs; the following does hold:

▶ **Proposition 2.2.** *For every connected digraph G with precisely one source node, there exists a TRS R and a term t of that TRS such that $\mathcal{G}(t) = G$. If G is finite, then R may be chosen to have finitely many rules.*

Proof. Let the signature of R have a distinct, nullary function symbol for each node of G, and for any two function symbols a and b let there be a rule $a \to b$ iff the node corresponding to a has a directed edge to the node corresponding to b. Let t be the function symbol corresponding to the source node of G. Then, $\mathcal{G}(t) = G$. ◀

Clearly, by the above proposition, Anagopos requires general graph drawing algorithms to draw reduction graphs.

For lambda calculus, we have:

▶ **Proposition 2.3.** *The following (families of) graphs are realizable as the underlying undirected versions of reduction graphs of lambda terms:*

1. *For every positive natural number n, the complete undirected graph K_n on n nodes.*
2. *For every positive natural number n, the underlying undirected graph of the n-dimensional hypercube.*

Proof. We give the families of terms explicitly (see also Figure 2):

1. Fix a variable y and define $k_1 = y$, and $k_{n+1} = (\lambda x.y)\, k_n$ for all $n \geq 1$. Then, for any $n \geq 1$, a straightforward induction shows that for all $1 \leq i \leq n$: $k_{n+1} \to_\beta k_i$, whence the result follows.
2. Fix a variable y and define $s = (\lambda x.x)\, y$. Set $s^1 = s$ and $s^n = s\, s^{n-1}$ for $n > 1$. Then the set of nodes of the reduction graph of s^n are all of the form $u_1 \cdots u_n$ with $u_i \in \{s, y\}$ for all $1 \leq i \leq n$, and directed edges between all pairs $u_1 \cdots u_{i-1}\, s\, u_{i+1} \cdots u_n$ and $u_1 \cdots u_{i-1}\, y\, u_{i+1} \cdots u_n$. Clearly, the underlying undirected graph is (isomorphic) to the n-dimensional hypercube. ◀

By the above, K_5 may occur as the underlying undirected graphs of lambda terms. Thus, by Kuratowski's theorem, reduction graphs of lambda terms are not, in general, planar.

Not every digraph can occur as reduction graph in lambda calculus; an example is the digraph $\bullet \longleftarrow \bullet \longrightarrow \bullet$, which cannot occur by the Church-Rosser theorem.

In addition to the above, observe that a reduction graph need not have a lattice structure, even in case of orthogonal systems [4, 18]: Consider, for example, the reduction graph of the lambda term $I((\lambda y.Ix)z)$ in Figure 3, taken from [19] and with $I = \lambda x.x$, where the starting term occurs at the bottom; note that the set consisting of the two nodes with two successors each does not have a least upper bound.

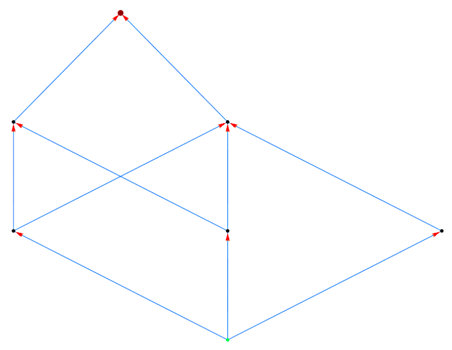


Figure 3 The graph of $I((\lambda y.Ix)z)$, as drawn with Dot (see Section 4). The starting term occurs at the bottom.

For lambda calculus and orthogonal term rewriting systems, one can instead consider another type of graph: The Hasse diagram of *reductions* with extension as ordering. As complete developments in such systems satisfy the Cube Lemma, the resulting graph is a complete lattice [4, 18]. Anagopos currently does not support visualization of these reductions.

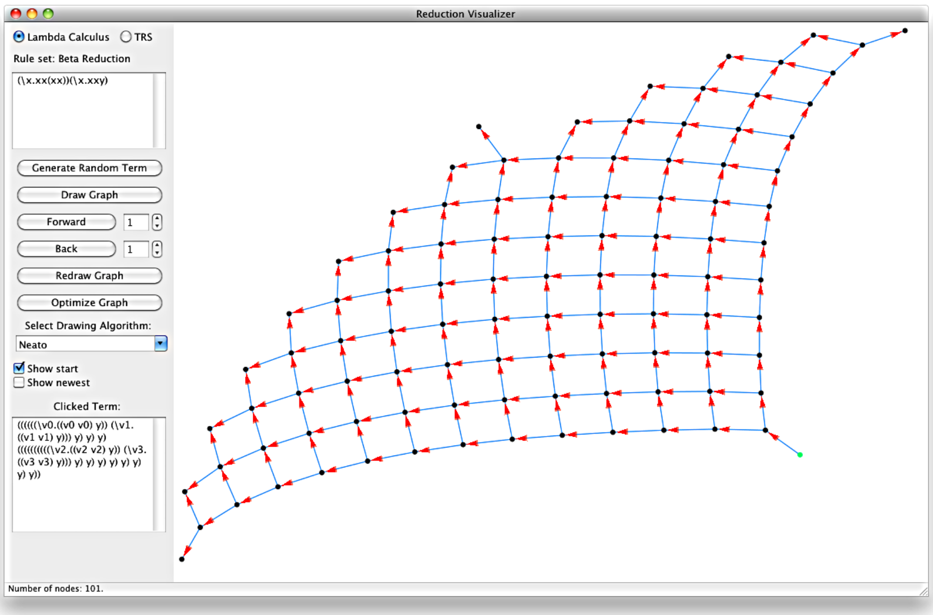
3 Anagopos—Interface and User's Guide

The main interface of Anagopos is shown in Figure 4. On the right, the reduction graph of the current term is shown (to improve aesthetic quality self-loops are omitted). The area on the left is divided into three distinct parts:

- At the top, see Figure 5(a), we can choose between visualization of either a lambda term or a first-order term and input the term we are interested in. The syntax of the terms is as expected, except that in lambda terms λ is replaced by \. A random term can also be generated.
- In the middle, see Figure 5(b), a number of buttons occur that influence the step-by-step drawing of reduction graphs, as explained below.
- At the bottom the term is displayed that corresponds to the last node from the reduction graph selected by the user.

Graph drawing—the middle-left area. The middle-left area supports the step-by-step drawing of reduction graphs. The *Draw Graph* button will display the nodes representing the initial term and its immediate reducts. Successors of reducts are, respectively, added and removed by pressing the *Forward* and *Backward* buttons, where the numbers indicate the number of reducts that will be treated.

Of the remaining interface elements, the *Redraw Graph* button resets the positions of the vertices and re-computes the layout of the reduction graph. The *Optimize Graph* button instructs the force-directed graph drawing algorithms (see Section 4) to attempt to improve the layout. Finally, the particular graph drawing algorithm can be selected under *Select Layout Algorithm* and *Show start* and *Show newest* mark, respectively, the initial vertex of the reduction graph and the last one whose immediate reducts were computed.



[✓ Lambda Calculus] TRS

Rule set: Beta Reduction

(\x.xx(xx))(\x.xxy)

Forward 1

Back 1

Select Drawing Algorithm: Neato

[✓ Show start]

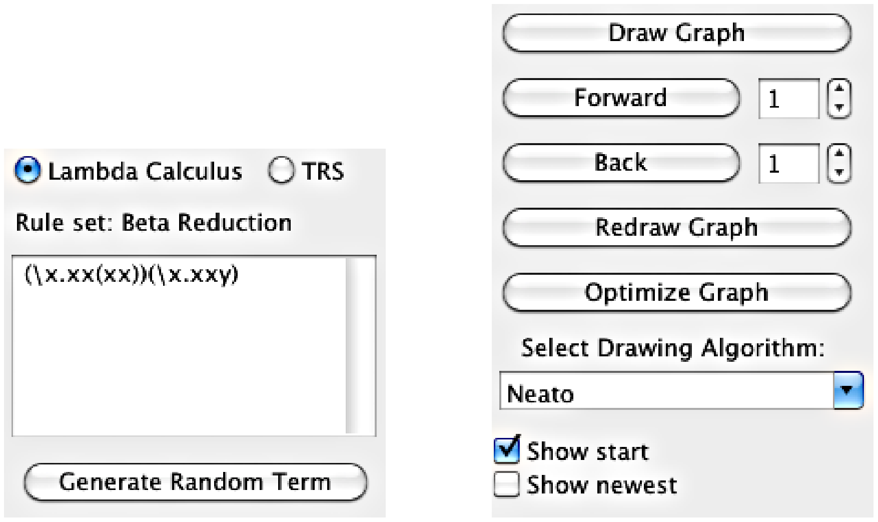
Show newest

Clicked Term:

```
((((((\v0.((v0 v0) y)) (\v1.
((v1 v1) y))) y) y) y)
((((((((((\v2.((v2 v2) y)) (\v3.
((v3 v3) y))) y) y) y) y) y) y)
y) y))
```

Number of nodes: 101.

Figure 4 Anagopos displaying part of $\mathcal{G}((\lambda x.xx(xx))(\lambda x.xxy))$ using Neato (see Section 4).



[✓ Lambda Calculus] TRS

Rule set: Beta Reduction

(\x.xx(xx))(\x.xxy)

Forward 1

Back 1

Select Drawing Algorithm: Neato

[✓ Show start]

Show newest

(a) The top-left area. **(b)** The middle-left area.

Figure 5 Close-ups of two parts of the user interface.

4 Anagopos—Graph Drawing Algorithms

Anagopos supports six general graph drawing algorithms, including three variations from the class of so-called *force-directed* algorithms, the current *de facto* standard in general graph drawing; all algorithms in this class draw graphs by employing minimization methods involving mechanical attraction and repulsion of nodes. For background on these drawing algorithms and others, including the ones mentioned below, we refer the reader to the relevant survey literature [3, 10, 14].

The six supported graph drawing algorithms are as follows, where we refer the reader to Figure 6 to get a taste of the drawings produced by each of the algorithms:

Neato and Neato Animated These are, respectively, the force-directed algorithm from [13, 8] and a slight variation with an animation-like appearance, which also draws the graph at intermediate stages of the minimization taking place.

Fdp This is an implementation of the force-directed graph drawing algorithm from [7].

Dot Constructs a hierarchical layout using Bézier curves for edges.

Circo and Twopi These are, respectively, an implementation of the algorithm from [28, 15], placing nodes on a circle, and the algorithm from [38], placing nodes on several concentric circles.

Implementation Details. Except for the Neato Animated algorithm, which we implemented ourselves to allow for its animation-like appearance, we draw heavily on the *GraphViz* graph visualization library [1], providing off-the-shelf implementations of all mentioned algorithms. This too explains the choice of the drawing algorithms.

To facilitate the implementation of the Neato Animated algorithm, we also implemented the highly efficient algorithm from [25] for solving the distance version of the all-pairs-shortest-path problem. As many graph drawing algorithms (and also other graph related tasks) depend on finding the graph-theoretical distance between all pairs of nodes, we expect this algorithm will find further uses in future extensions of Anagopos.

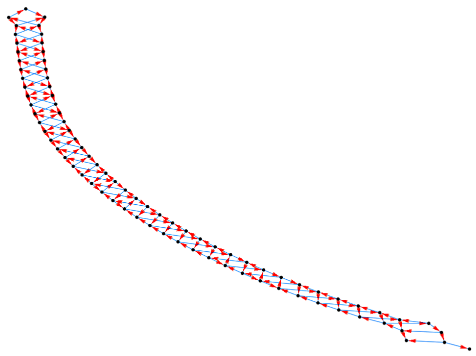
Anagopos lends itself to easy addition of new graph drawing algorithms; one is only required to implement a very simple interface `DrawingAlgorithm` through which (a) a graph can be passed to the drawing algorithm, and through which (b) the algorithm can be told to compute a layout for the given graph.

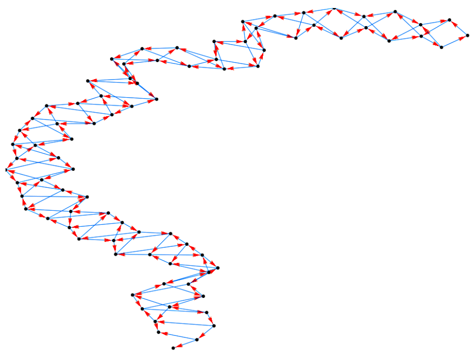
5 Anagopos—Architecture

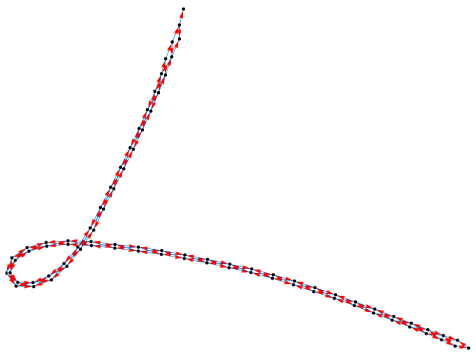
Anagopos is implemented in Python 2.6 [23], making heavy use of the object-oriented features of the language. The tool has a Model-View-Controller architecture [17], separating the model (the reduction graph of a term) from the manipulation of the model (user input) and the presentation of the model (display of the reduction graph).

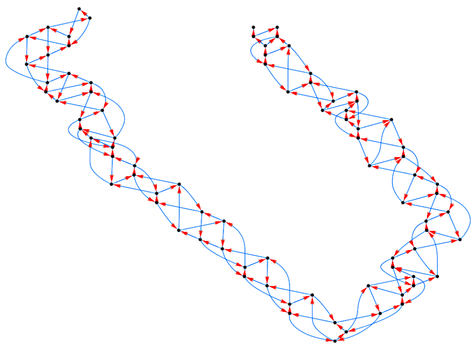
The interface is implemented using wxWidgets [29], a cross-platform GUI library. The various parsers of Anagopos (for parsing TRSs, first-order terms, and lambda terms), are simple and implemented either by hand or using pyparsing [22]. The only exception is the TPDB parser which is constructed around the Expat XML parser [6].

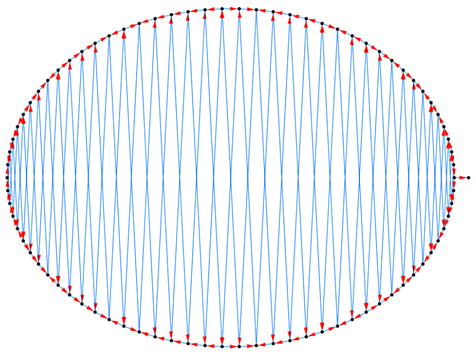
Internally, terms are represented as DAGs; bound variables of lambda terms are canonized to ensure alpha equivalent terms have a unique representation (canonization retains information on unbound variables within the term structure). Reduction graphs are represented as instances of a custom-made `Graph`-class.


(a) Neato: Attempts to ensure that neighboring nodes are at equal distance from each other.


(b) Neato Animated: As Neato, but with optimizations animated. The picture shows the graph at a stage where it is not fully optimized.


(c) Fdp: Produces a graph comparable to Neato, but in this case reaches a local minimum with a loop-like quality.


(d) Dot: Attempts to create a hierarchy, which proves difficult in this case due to the large number of cycles in the reduction graph.


(e) Circo: Places all nodes of the reduction graph on a circle.

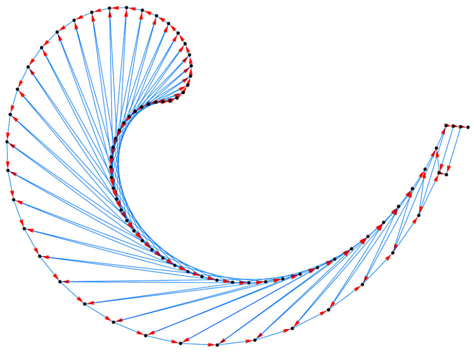
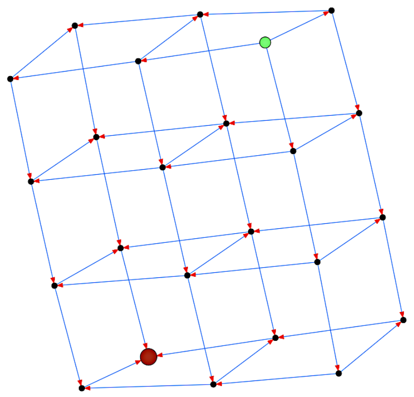

(f) Twopi: Places the nodes on several concentric circles.

Figure 6 A partial reduction graph of the lambda term HIH from [2, Exercise 3.5.5(i)], with $H = \lambda xy.x(\lambda z.yzy)x$ and $I = \lambda x.x$, as drawn with each of the supported drawing algorithms.


(a) The reduction graph of the term $((\lambda x.x)y)((\lambda x.xxx)\lambda y.y)((\lambda x.xx)\lambda y.y)$.

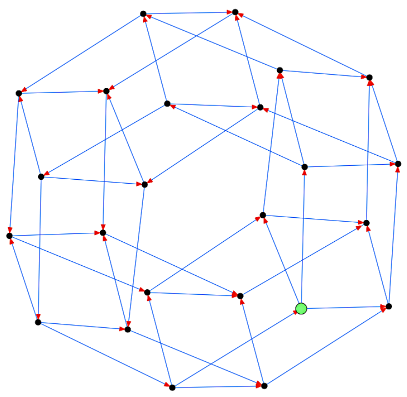

(b) $\mathcal{G}((ttI)c_2)$ with $t = \lambda xy.yyyyxxy$, $I = \lambda x.x$, and c_2 as in Proposition 2.3.

Figure 7 The "poor man's 3D-effect" visible in some of the reduction graphs drawn with Neato.

The reduction graphs are generated in a breadth-first fashion. Currently, Anagopos contracts at most 10^6 redexes per graph; this value is easily changed, but GraphViz performance becomes problematic for graphs of larger size.

6 Conclusion and Future Work

We have presented Anagopos, a tool for drawing reduction graphs of both lambda terms and terms from term rewriting. While we regard Anagopos in its current incarnation mainly as a tool for education and leisure, we believe that the tool may prove useful for hypothesis formation and, possibly, the formulation of proofs of new results concerning reduction graphs. As an appetizer, we mention the following problems:

- The set of reduction graphs of the set of terms of a TRS with finite signature is recursively enumerable. *Which recursively enumerable classes of finite graphs are realizable as the set of reduction graphs of a terminating TRS*? And, if we drop the requirement that graphs need to be finite, *which classes of countable graphs are realizable as the set of reduction graphs of a TRS*?
- *Precisely which* un*directed graphs can be realized as the underlying undirected graphs of lambda terms*?

Anagopos is open source and easily extended. Based on our initial experimentation with the tool, it would be obvious to include support for some general higher-order rewriting format and to implement zooming, panning, and manual rearrangements of the nodes of graphs. Furthermore, considering the tantalizing "poor man's 3D-effect" of force-directed algorithms such as Neato (see Figure 7), it seems natural to try to generate three-dimensional representations of the graphs combined with rotation. Moreover, use could be made of a general-purpose visualization framework like Tulip [32], which not only allows for easy visualization of graphs, but also of meta-data concerning the graphs. Finally, while we have considered only the most basic notion of a reduction graph, as induced by the "raw" rewrite relation of lambda calculus and TRSs, the standard literature on rewriting also considers equivalence relations over reductions (e.g., in orthogonal TRSs) [30, Chapter 8]; it would be natural if Anagopos could also take an equivalence relation as input and generate the resulting graphs. This could include equivalence relations designed specifically to give insight into the global structure of reduction graphs; something which has already shown its usefulness in the area of state space visualization [9].

Acknowledgments The authors thank Jan Willem Klop, Peter Sestoft, and the anonymous referees for useful feedback.

References

1 AT&T Research. Graphviz – graph visualization software. Available from: `http://www.graphviz.org/`. Accessed 31 December 2010.

2 H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* Elsevier Science, revised edition, 1984. First edition 1981.

3 G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry: Theory and Applications*, 4(5):235–282, 1994.

4 G. Berry and J.-J. Lévy. Minimal and optimal computations of recursive programs. *Journal of the ACM*, 26:148–175, 1979.

5 J. Endrullis. Graphical lambda calculator. Available from: `http://joerg.endrullis.de/lambdaCalculator.html`. Accessed 31 December 2010.

6 Expat maintainers. The Expat XML parser library. Available from: `http://expat.sourceforge.net/`. Accessed 5 January 2011.

7 T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software – Practice and Experience*, 21(11):1129–1164, 1991.

8 E. R. Gansner, Y. Koren, and S. North. Graph Drawing by Stress Majorization. In J. Pach, editor, *Proceedings of the 12th International Symposium on Graph Drawing (GD 2004)*, volume 3383 of *Lecture Notes in Computer Science*, pages 239–250, 2004.

9 J. F. Groote and F. van Ham. Interactive visualization of large state spaces. *International Journal on Software Tools for Technology Transfer*, 8(1):77–91, 2006.

10 I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: a survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(10):24–43, 2000.

11 B. Intrigila and A. R. Laurenzi. Two problems on reduction graphs in lambda calculus. *Fundamenta Informaticae*, 44(1-2):133–144, 2000.

12 B. Intrigila and M. Venturini Zilli. Orders, reduction graphs and spectra. *Theoretical Computer Science*, 212(1-2):211–231, 1999.

13 T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989.

14 M. Kaufmann and D. Wagner, editors. *Drawing Graphs: Methods and Models*, volume 2025 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

15 M. Kaufmann and R. Wiese. Maintaining the mental map for circular drawings. In S. G. Kobourov and M. T. Goodrich, editors, *Proceedings of the 10th International Symposium on Graph Drawing (GD 2002)*, volume 2528 of *Lecture Notes in Computer Science*, pages 12–22, 2002.

16 J. W. Klop. Reduction cycles in combinatory logic. In J. Seldin and J. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 193–214. Academic Press, 1980.

17 G. E. Krasner and S. T. Pope. A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80. *Journal of Object Oriented Programming*, 1:26–49, August/September 1988.

18 J.-J. Lévy. *Réductions correctes et optimales dans le lambda-calcul.* PhD thesis, Université de Paris VII, 1978.

19 J.-J. Lévy. Generalized finite developments. In Y. Bertot, G. Huet, J.-J. Lévy, and G. Plotkin, editors, *From Semantics to Computer Science: Essys in Honour of Gilles Kahn*, pages 185–204. Cambridge University Press, 2009.

20 S. Lippi. in^2 : A graphical interpreter for interaction nets. In S. Tison, editor, *Proceedings of the 13th International Conference on Rewriting Techniques and Applications (RTA 2002)*, volume 2378 of *Lecture Notes in Computer Science*, pages 380–386, 2002.

21 J. Matthews, R. B. Findler, M. Flatt, and M. Felleisen. A visual environment for developing context-sensitive term rewriting systems. In V. van Oostrom, editor, *Proceedings of the 15th International Conference on Rewriting Techniques and Applications (RTA 2004)*, volume 3091 of *Lecture Notes in Computer Science*, pages 301–311, 2004.

22 P. McGuire. pyparsing. Available from: `http://pyparsing.wikispaces.com/`. Accessed 5 January 2011.

23 Python Software Foundation. Python programming language. Available from: `http://www.python.org/`. Accessed 4 January 2011.

24 D. Ruiz and M. Villaret. TILC: The Interactive Lambda-Calculus Tracer. *Electronic Notes in Theoretical Computer Science*, 248:173–183, 2009.

25 R. Seidel. On the all-pairs-shortest-path problem. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC'92)*, pages 745–749, 1992.

26 S. Sekimoto and S. Hirokawa. One-step recurrent terms in lambda-beta-calculus. *Theoretical Computer Science*, 56:223–231, 1988.

27 P. Sestoft. Standard ML on the web server: Visualizing lambda calculus reduction. Technical report, Royal Veterinary and Agricultural University, Denmark, 1996.

28 J. M. Six and I. G. Tollis. A framework for circular drawings of networks. In J. Kratochvíl, editor, *Proceedings of the 7th International Symposium on Graph Drawing (GD'99)*, volume 1731 of *Lecture Notes in Computer Science*, pages 107–116, 1999.

29 J. Smart, R. Roebling, et al. wxWidgets – cross-platform GUI library. Available from: `http://www.wxwidgets.org/`. Accessed 5 January 2011.

30 Terese, editor. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

31 M. Thyer. Lambda Animator: animated reduction of the lambda calculus. Available from: `http://thyer.name/lambda-animator/`. Accessed 31 December 2010.

32 University of Bordeaux I. Tulip: Data Visualization Software. Available from: `http://www.tulip-software.org/`. Accessed 28 March 2011.

33 D. van den Eijkel. Animated Lambda Calculus Evaluator (ALCE). Available from: `http://substitut-fuer-feinmotorik.net/projects/animated-lambda-calculus-evaluator`. Accessed 31 December 2010.

34 M. Venturini Zilli. Cofinality in reduction graphs. In G. Ausiello and M. Protasi, editors, *Proceedings of the 8th Colloquium on Trees in Algebra and Programming (CAAP'83)*, volume 159 of *Lecture Notes in Computer Science*, pages 405–416, 1983.

35 M. Venturini Zilli. Reduction graphs in the lambda calculus. *Theoretical Computer Science*, 29:251–275, 1984.

36 B. Victor. Alligator Eggs – a puzzle game. Available from: `http://worrydream.com/AlligatorEggs/`. Accessed 31 December 2010.

37 F. Wiedijk. Untyped lambda calculus. Available from: `http://www.cs.ru.nl/~freek/notes/lambda.ml`. Accessed 31 December 2010.

38 G. J. Wills. NicheWorks – interactive visualization of very large graphs. In G. D. Battista, editor, *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 403–414, 1997.