Vincent van Oostrom (Ed.)

LNCS 3091

Rewriting Techniques and Applications

15th International Conference, RTA 2004
Aachen, Germany, June 2004
Proceedings

Springer

Vincent van Oostrom (Ed.)

Rewriting Techniques and Applications

15th International Conference, RTA 2004
Aachen, Germany, June 3-5, 2004
Proceedings

Springer

Volume Editor

Vincent van Oostrom
Universiteit Utrecht, Department of Philosophy
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands
E-mail: Vincent.vanOostrom@phil.uu.nl

Library of Congress Control Number: 2004106912

CR Subject Classification (1998): F.4, F.3.2, D.3, I.2.2-3, I.1

ISSN 0302-9743
ISBN 3-540-22153-0 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 11011743 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3091

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Preface

This volume contains the proceedings of the *15th International Conference on Rewriting Techniques and Applications* (RTA 2004), which was held June 2–5, 2004, at the RWTH Aachen in Germany. RTA is the major forum for the presentation of research on all aspects of rewriting. Previous RTA conferences took place in Dijon (1985), Bordeaux (1987), Chapel Hill (1989), Como (1991), Montreal (1993), Kaiserslautern (1995), Rutgers (1996), Sitges (1997), Tsukuba (1998), Trento (1999), Norwich (2000), Utrecht (2001), Copenhagen (2002), and Valencia (2003).

The program committee selected 19 papers for presentation, including five system descriptions, from a total of 43 submissions. In addition, there were invited talks by Neil Jones, Aart Middeldorp, and Robin Milner.

Many people helped to make RTA 2004 a success. I am grateful to the members of the program committee and the external referees for reviewing the submissions and maintaining the high standards of the RTA conferences. It is a great pleasure to thank the conference chair Jürgen Giesl and the other members of the local organizing committee. They were in charge of the local organization of all events partaking in the Federated Conference on Rewriting, Deduction, and Programming (RDP). Apart from RTA 2004, these events were:

- 2nd International Workshop on Higher-Order Rewriting (Delia Kesner, Femke van Raamsdonk, and Joe Wells),
- 5th International Workshop on Rule-Based Programming (Slim Abdennadher and Christophe Ringeissen),
- 13th International Workshop on Functional and (Constraint) Logic Programming (Herbert Kuchen),
- IFIP Working Group 1.6 on Term Rewriting (Claude Kirchner),
- 4th International Workshop on Reduction Strategies in Rewriting and Programming (Sergio Antoy and Yoshihito Toyama),
- 7th International Workshop on Termination (Michael Codish and Aart Middeldorp).

I thank the organizers of all these events for making RTA 2004 more attractive by collocating with it. Finally, I gratefully acknowledge the financial support of the Deutsche Forschungsgemeinschaft (DFG).

April 2004 Vincent van Oostrom

Conference Organization

Program Chair

Vincent van Oostrom	*Universiteit Utrecht*

Conference Chair

Jürgen Giesl	*RWTH Aachen*

Program Committee

Zena Ariola	*Oregon*
Jürgen Giesl	*Aachen*
Masahito Hasegawa	*Kyoto*
Hélène Kirchner	*Nancy*
Pierre Lescanne	*Lyon*
Klaus Madlener	*Kaiserslautern*
Narciso Martí-Oliet	*Madrid*
Paul-André Melliès	*Paris*
Oege de Moor	*Oxford*
Vincent van Oostrom	*Utrecht*
Frank Pfenning	*Carnegie Mellon*
Ashish Tiwari	*SRI*
Ralf Treinen	*ENS Cachan*
Roel de Vrijer	*Amsterdam*

Organizing Committee

Jürgen Giesl
Elke Ohlenforst
Arnd Gehrmann
Peter Schneider-Kamp
René Thiemann

RTA Steering Committee

Franz Baader	*Dresden*	
Pierre Lescanne	*Lyon*	
Aart Middeldorp	*Innsbruck*	(chair)
Femke van Raamsdonk	*Amsterdam*	(publicity chair)
Robert Nieuwenhuis	*Barcelona*	
Ralf Treinen	*Cachan*	

List of Referees

Andreas Abel
Jürgen Avenhaus
Franco Barbanera
Alessandro Berarducci
Stefan Blom
Olivier Bournez
Patricia Bouyer
Mark van den Brand
Thierry Cachat
Adam Cichon
Horatiu Cirstea
Dan Dougherty
Steven Eker
Olivier Fissore
David de Frutos-Escrig
Thomas Genet
Alfons Geser
Silvia Ghilezan
Isabelle Gnaedig
Guillem Godoy Balil
Gunter Grieser
Makoto Hamana
Thomas Hillenbrand
Daniel Hirschkoff
Petra Hofstedt
Florent Jacquemard
Patricia Johann
Thierry Joly
Neil Jones
Deepak Kapur
Delia Kesner
Jeroen Ketema
Sava Krstic
Frédéric Lang
Sébastien Limet
Denis Lugiez
Claude Marché
Maurice Margenstern
Ralph Matthes
Aart Middeldorp
Pierre-Etienne Moreau
Paliath Narendran
Monica Nesi
Joachim Niehren
Tobias Nipkow
Hitoshi Ohsaki
Nicolas Ollinger
Peter Ölveczky
Friedrich Otto
Miguel Palomino
Ricardo Peña Marí
Brigitte Pientka
Leonor Prensa Nieto
Maurizio Proietti
Femke van Raamsdonk
Christophe Ringeissen
Mario Rodríguez-Artalejo
Andrea Sattler-Klein
Manfred Schmidt-Schauss
Klaus Schneider
Peter Schneider-Kamp
Helmut Seidl
Damien Sereni
Natarajan Shankar
Ganesh Sittampalam
Mark-Oliver Stehr
Toshinori Takai
Bas Terwijn
René Thiemann
Yoshihito Toyama
Akihiko Tozawa
Xavier Urbain
Rakesh Verma
Laurent Vigneron
Eelco Visser
Fer-Jan de Vries
Clemens Grabmayer
Benjamin Wack
Martijn Warnier
Rolf Wiehagen
Hans Zantema

Author Index

Lecture Notes in Computer Science

For information about Vols. 1–2977

please contact your bookseller or Springer-Verlag

Vol. 3092: J. Eckstein, H. Baumeister (Eds.), Extreme Programming and Agile Processes in Software Engineering. XVI, 358 pages. 2004.

Vol. 3091: V. van Oostrom (Ed.), Rewriting Techniques and Applications. X, 313 pages. 2004.

Vol. 3084: A. Persson, J. Stirna (Eds.), Advanced Information Systems Engineering. XIV, 596 pages. 2004.

Vol. 3083: W. Emmerich, A.L. Wolf (Eds.), Component Deployment. X, 249 pages. 2004.

Vol. 3076: D. Buell (Ed.), Algorithmic Number Theory. XI, 451 pages. 2004.

Vol. 3074: B. Kuijpers, P. Revesz (Eds.), Constraint Databases and Applications. XII, 181 pages. 2004.

Vol. 3066: S. Tsumoto, R. Słowiński, J. Komorowski, J.W. Grzymala-Busse (Eds.), Rough Sets and Current Trends in Computing. XX, 853 pages. 2004. (Subseries LNAI).

Vol. 3065: A. Lomuscio, D. Nute (Eds.), Deontic Logic. X, 275 pages. 2004. (Subseries LNAI).

Vol. 3064: D. Bienstock, G. Nemhauser (Eds.), Integer Programming and Combinatorial Optimization. XI, 445 pages. 2004.

Vol. 3063: A. Llamosí, A. Strohmeier (Eds.), Reliable Software Technologies - Ada-Europe 2004. XIII, 333 pages. 2004.

Vol. 3062: J.L. Pfaltz, M. Nagl, B. Böhlen (Eds.), Applications of Graph Transformations with Industrial Relevance. XV, 500 pages. 2004.

Vol. 3060: A.Y. Tawfik, S.D. Goodwin (Eds.), Advances in Artificial Intelligence. XIII, 582 pages. 2004. (Subseries LNAI).

Vol. 3059: C.C. Ribeiro, S.L. Martins (Eds.), Experimental and Efficient Algorithms. X, 586 pages. 2004.

Vol. 3058: N. Sebe, M.S. Lew, T.S. Huang (Eds.), Computer Vision in Human-Computer Interaction. X, 233 pages. 2004.

Vol. 3056: H. Dai, R. Srikant, C. Zhang (Eds.), Advances in Knowledge Discovery and Data Mining. XIX, 713 pages. 2004. (Subseries LNAI).

Vol. 3054: I. Crnkovic, J.A. Stafford, H.W. Schmidt, K. Wallnau (Eds.), Component-Based Software Engineering. XI, 311 pages. 2004.

Vol. 3053: C. Bussler, J. Davies, D. Fensel, R. Studer (Eds.), The Semantic Web: Research and Applications. XIII, 490 pages. 2004.

Vol. 3052: W. Zimmermann, B. Thalheim (Eds.), Abstract State Machines 2004. Advances in Theory and Practice. XII, 235 pages. 2004.

Vol. 3051: R. Berghammer, B. Möller, G. Struth (Eds.), Relational and Kleene-Algebraic Methods in Computer Science. X, 279 pages. 2004.

Vol. 3047: F. Oquendo, B. Warboys, R. Morrison (Eds.), Software Architecture. X, 279 pages. 2004.

Vol. 3046: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications - ICCSA 2004. LIII, 1016 pages. 2004.

Vol. 3045: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications – ICCSA 2004. LIII, 1040 pages. 2004.

Vol. 3044: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications – ICCSA 2004. LIII, 1140 pages. 2004.

Vol. 3043: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.J.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications – ICCSA 2004. LIII, 1180 pages. 2004.

Vol. 3042: N. Mitrou, K. Kontovasilis, G.N. Rouskas, I. Iliadis, L. Merakos (Eds.), NETWORKING 2004, Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications. XXXIII, 1519 pages. 2004.

Vol. 3039: M. Bubak, G.D.v. Albada, P.M. Sloot, J. Dongarra (Eds.), Computational Science - 2004. LXVI, 1271 pages. 2004.

Vol. 3038: M. Bubak, G.D.v. Albada, P.M. Sloot, J. Dongarra (Eds.), Computational Science - ICCS 2004. LXVI, 1311 pages. 2004.

Vol. 3037: M. Bubak, G.D.v. Albada, P.M. Sloot, J. Dongarra (Eds.), Computational Science - ICCS 2004. LXVI, 745 pages. 2004.

Vol. 3036: M. Bubak, G.D.v. Albada, P.M. Sloot, J. Dongarra (Eds.), Computational Science - ICCS 2004. LXVI, 713 pages. 2004.

Vol. 3035: M.A. Wimmer (Ed.), Knowledge Management in Electronic Government. XII, 326 pages. 2004. (Subseries LNAI).

Vol. 3034: J. Favela, E. Menasalvas, E. Chávez (Eds.), Advances in Web Intelligence. XIII, 227 pages. 2004. (Subseries LNAI).

Vol. 3033: M. Li, X.-H. Sun, Q. Deng, J. Ni (Eds.), Grid and Cooperative Computing. XXXVIII, 1076 pages. 2004.

Vol. 3032: M. Li, X.-H. Sun, Q. Deng, J. Ni (Eds.), Grid and Cooperative Computing. XXXVII, 1112 pages. 2004.

Vol. 3031: A. Butz, A. Krüger, P. Olivier (Eds.), Smart Graphics. X, 165 pages. 2004.

Vol. 3030: P. Giorgini, B. Henderson-Sellers, M. Winikoff (Eds.), Agent-Oriented Information Systems. XIV, 207 pages. 2004. (Subseries LNAI).

Vol. 3029: B. Orchard, C. Yang, M. Ali (Eds.), Innovations in Applied Artificial Intelligence. XXI, 1272 pages. 2004. (Subseries LNAI).

Vol. 3028: D. Neuenschwander, Probabilistic and Statistical Methods in Cryptology. X, 158 pages. 2004.

Vol. 3027: C. Cachin, J. Camenisch (Eds.), Advances in Cryptology - EUROCRYPT 2004. XI, 628 pages. 2004.

Vol. 3026: C. Ramamoorthy, R. Lee, K.W. Lee (Eds.), Software Engineering Research and Applications. XV, 377 pages. 2004.

Vol. 3025: G.A. Vouros, T. Panayiotopoulos (Eds.), Methods and Applications of Artificial Intelligence. XV, 546 pages. 2004. (Subseries LNAI).

Vol. 3024: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 621 pages. 2004.

Vol. 3023: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 611 pages. 2004.

Vol. 3022: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 621 pages. 2004.

Vol. 3021: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 633 pages. 2004.

Vol. 3019: R. Wyrzykowski, J. Dongarra, M. Paprzycki, J. Wasniewski (Eds.), Parallel Processing and Applied Mathematics. XIX, 1174 pages. 2004.

Vol. 3016: C. Lengauer, D. Batory, C. Consel, M. Odersky (Eds.), Domain-Specific Program Generation. XII, 325 pages. 2004.

Vol. 3015: C. Barakat, I. Pratt (Eds.), Passive and Active Network Measurement. XI, 300 pages. 2004.

Vol. 3014: F. van der Linden (Ed.), Software Product-Family Engineering. IX, 486 pages. 2004.

Vol. 3012: K. Kurumatani, S.-H. Chen, A. Ohuchi (Eds.), Multi-Agnets for Mass User Support. X, 217 pages. 2004. (Subseries LNAI).

Vol. 3011: J.-C. Régin, M. Rueher (Eds.), Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. XI, 415 pages. 2004.

Vol. 3010: K.R. Apt, F. Fages, F. Rossi, P. Szeredi, J. Váncza (Eds.), Recent Advances in Constraints. VIII, 285 pages. 2004. (Subseries LNAI).

Vol. 3009: F. Bomarius, H. Iida (Eds.), Product Focused Software Process Improvement. XIV, 584 pages. 2004.

Vol. 3008: S. Heuel, Uncertain Projective Geometry. XVII, 205 pages. 2004.

Vol. 3007: J.X. Yu, X. Lin, H. Lu, Y. Zhang (Eds.), Advanced Web Technologies and Applications. XXII, 936 pages. 2004.

Vol. 3006: M. Matsui, R. Zuccherato (Eds.), Selected Areas in Cryptography. XI, 361 pages. 2004.

Vol. 3005: G.R. Raidl, S. Cagnoni, J. Branke, D.W. Corne, R. Drechsler, Y. Jin, C.G. Johnson, P. Machado, E. Marchiori, F. Rothlauf, G.D. Smith, G. Squillero (Eds.), Applications of Evolutionary Computing. XVII, 562 pages. 2004.

Vol. 3004: J. Gottlieb, G.R. Raidl (Eds.), Evolutionary Computation in Combinatorial Optimization. X, 241 pages. 2004.

Vol. 3003: M. Keijzer, U.-M. O'Reilly, S.M. Lucas, E. Costa, T. Soule (Eds.), Genetic Programming. XI, 410 pages. 2004.

Vol. 3002: D.L. Hicks (Ed.), Metainformatics. X, 213 pages. 2004.

Vol. 3001: A. Ferscha, F. Mattern (Eds.), Pervasive Computing. XVII, 358 pages. 2004.

Vol. 2999: E.A. Boiten, J. Derrick, G. Smith (Eds.), Integrated Formal Methods. XI, 541 pages. 2004.

Vol. 2998: Y. Kameyama, P.J. Stuckey (Eds.), Functional and Logic Programming. X, 307 pages. 2004.

Vol. 2997: S. McDonald, J. Tait (Eds.), Advances in Information Retrieval. XIII, 427 pages. 2004.

Vol. 2996: V. Diekert, M. H ıbib (Eds.), STACS 2004. XVI, 658 pages. 2004.

Vol. 2995: C. Jensen, S. Poslad, T. Dimitrakos (Eds.), Trust Management. XIII, 377 pages. 2004.

Vol. 2994: E. Rahm (Ed.), Data Integration in the Life Sciences. X, 221 pages. 2004. (Subseries LNBI).

Vol. 2993: R. Alur, G.J. Pappas (Eds.), Hybrid Systems: Computation and Control. XII, 674 pages. 2004.

Vol. 2992: E. Bertino, S. Christodoulakis, D. Plexousakis, V. Christophides, M. Koubarakis, K. Böhm, E. Ferrari (Eds.), Advances in Database Technology - EDBT 2004. XVIII, 877 pages. 2004.

Vol. 2991: R. Alt, A. Frommer, R.B. Kearfott, W. Luther (Eds.), Numerical Software with Result Verification. X, 315 pages. 2004.

Vol. 2990: J. Leite, A. Omicini, L. Sterling, P. Torroni (Eds.), Declarative Agent Languages and Techniques. XII, 281 pages. 2004. (Subseries LNAI).

Vol. 2989: S. Graf, L. Mounier (Eds.), Model Checking Software. X, 309 pages. 2004.

Vol. 2988: K. Jensen, A. Podelski (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. XIV, 608 pages. 2004.

Vol. 2987: I. Walukiewicz (Ed.), Foundations of Software Science and Computation Structures. XIII, 529 pages. 2004.

Vol. 2986: D. Schmidt (Ed.), Programming Languages and Systems. XII, 417 pages. 2004.

Vol. 2985: E. Duesterwald (Ed.), Compiler Construction. X, 313 pages. 2004.

Vol. 2984: M. Wermelinger, T. Margaria-Steffen (Eds.), Fundamental Approaches to Software Engineering. XII, 389 pages. 2004.

Vol. 2983: S. Istrail, M.S. Waterman, A. Clark (Eds.), Computational Methods for SNPs and Haplotype Inference. IX, 153 pages. 2004. (Subseries LNBI).

Vol. 2982: N. Wakamiya, M. Solarski, J. Sterbenz (Eds.), Active Networks. XI, 308 pages. 2004.

Vol. 2981: C. Müller-Schloer, T. Ungerer, B. Bauer (Eds.), Organic and Pervasive Computing – ARCS 2004. XI, 339 pages. 2004.

Vol. 2980: A. Blackwell, K. Marriott, A. Shimojima (Eds.), Diagrammatic Representation and Inference. XV, 448 pages. 2004. (Subseries LNAI).

Vol. 2979: I. Stoica, Stateless Core: A Scalable Approach for Quality of Service in the Internet. XVI, 219 pages. 2004.

Vol. 2978: R. Groz, R.M. Hierons (Eds.), Testing of Communicating Systems. XII, 225 pages. 2004.

Table of Contents

Termination Analysis of the Untyped λ-Calculus

Neil D. Jones[1] and Nina Bohr[2]

[1] DIKU, University of Copenhagen
[2] IT University of Copenhagen

Abstract. An algorithm is developed that, given an untyped λ-expression, can certify that its call-by-value evaluation will terminate. It works by an extension of the "size-change principle" earlier applied to first-order programs. The algorithm is sound (and proven so in this paper) but not complete: some λ-expressions may in fact terminate under call-by-value evaluation, but not be recognised as terminating.
The *intensional* power of size-change termination is reasonably high: It certifies as terminating all primitive recursive programs, and many interesting and useful general recursive *algorithms* including programs with mutual recursion and parameter exchanges, and Colson's "minimum" algorithm. Further, the approach allows free use of the Y combinator, and so can identify as terminating a substantial subset of PCF.
The *extensional power* of size-change termination is the set of functions computable by size-change terminating programs. This lies somewhere between Péter's multiple recursive functions and the class of ϵ_0-recursive functions.

1 Introduction

The *size-change* analysis of [5] can show termination of first-order functional programs whose parameter values have a well-founded size order. The method is reasonably general, easily automated, and does not require human invention of lexical or other parameter orders. This paper applies similar ideas to establish termination of *higher-order* programs. For simplicity and generality we focus on the simplest such language, the λ-calculus. We expect that the framework can be naturally extended to higher-order functional programs, e.g., functional subsets of Scheme or ML.

1.1 An Example of Size-Change Analysis

Example 1. A motivating example is the size-change termination analysis of a first-order program, using the framework of [5]. Consider a program with functions `f` and `g` defined by mutual recursion:

```
f(x,y)   = if x=0 then y else 1: g(x,y,y)
g(u,v,w) = if w=0 then 3:f(u-1,w) else 2:g(u,v-1,w+2)
```

The three function calls have been labeled 1, 2 and 3. The "control flow graph" in Figure 1 shows the calling function and called function of each call, e.g.,

V. van Oostrom (Ed.): RTA 2004, LNCS 3091, pp. 1–23, 2004.

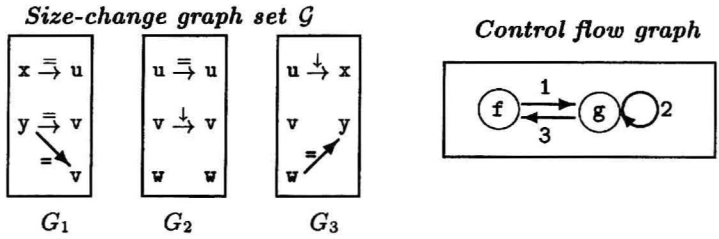

Fig. 1. Call graph and size-change graphs for the example first-order program.

$3 : g \to f$. Each call is associated with a "size-change graph", e.g., G_3 for call 3, that describes the data flow from the calling function's parameters to the called function's parameters.

Termination reasoning: Consider (in order to prove it impossible) any *infinite size-change graph sequence* $\mathcal{M} = g_1 g_2 \ldots \in \{G_1, G_2, G_3\}^\omega$ that follows the program's control flow:

Case 1: $\mathcal{M} = \ldots (G_2)^\omega$ ends in infinitely many G_2's: In this case, *variable* v *descends infinitely.*

Case 2: $\mathcal{M} = \ldots (G_1 G_2^* G_3)^\omega$. In this case, *variable* u *descends infinitely.*

Both cases are impossible (assuming, as we do, that the data value set is well-founded). Therefore a call of *any program function with any data will terminate.*

End of example.

1.2 Definitions and Terminology

We describe the structure of size-change graphs and state the size-change termination condition.

Definition 1.

1. *A* size-change graph $A \xrightarrow{G} B$ *consists of a* source *set A; a* target *set B; and a set of labeled arcs* $G \subseteq A \times \{=, \downarrow\} \times B$. *If* A, B *are clear from context, we identify* $A \xrightarrow{G} B$ *with its arc set G.*
2. *The* identity *size-change graph for A is* $A \xrightarrow{id_A} A$ *where* $id_A = \{x \stackrel{=}{\to} x \mid x \in A\}$.
3. *Size-change graphs* $A \xrightarrow{G_1} B$ *and* $C \xrightarrow{G_2} D$ *are* composible *if* $B = C$.
4. *The* sequential composition *of size-change graphs* $A \xrightarrow{G_1} B$ *and* $B \xrightarrow{G_2} C$ *is* $A \xrightarrow{G_1;G_2} C$ *where*

$$G_1; G_2 = \Big\{ x \xrightarrow{\downarrow} z \mid \downarrow \in \{r, s \mid x \xrightarrow{r} y \in G_1 \text{ and } y \xrightarrow{s} z \in G_2 \text{ for some } y \in B\} \Big\}$$
$$\cup \Big\{ x \stackrel{=}{\to} z \mid \{=\} = \{r, s \mid x \xrightarrow{r} y \in G_1 \text{ and } y \xrightarrow{s} z \in G_2 \text{ for some } y \in B\} \Big\}$$

Lemma 1. *Sequential composition is associative.* $A \xrightarrow{G} B$ *implies* $id_A; G = G; id_B = G$.

Definition 2. *A* multipath $\mathcal{M}$ *over a set* $\mathcal{G}$ *of size-change graphs is a finite or infinite composible sequence of graphs in* $\mathcal{G}$. *Define*

$$\mathcal{G}^\omega = \{\mathcal{M} = G_0, G_1, \ldots \mid \text{graphs } G_i, G_{i+1} \text{ are composible for } i = 0, 1, 2, \ldots\}$$

Definition 3.

1. *A* thread *in a multipath* $\mathcal{M} = G_0, G_1, G_2, \ldots$ *is a sequence* $t = a_j \xrightarrow{r_j} a_{j+1} \xrightarrow{r_{j+1}} \ldots$ *such that* $a_k \xrightarrow{r_k} a_{k+1} \in G_k$ *for every* $k \geq j$ *(and each* r_k *is* $= \text{or} \downarrow$.*)*
2. *Thread* t *is of* infinite descent *if* $r_k = \downarrow$ *for infinitely many* $k \geq j$.
3. *A set* $\mathcal{G}$ *of size-change graphs satisfies the* size-change condition *if every* $\mathcal{M} \in \mathcal{G}^\omega$ *contains at least one thread of infinite descent.*

The size-change condition is decidable; its worst-case complexity, as a function of the size of the program being analysed, is shown complete for PSPACE in [5].

The example revisited: The program of Figure 1 has three size-change graphs, one for each of the calls $\mathtt{1} : \mathtt{f} \to \mathtt{g}, \mathtt{2} : \mathtt{g} \to \mathtt{g}, \mathtt{3} : \mathtt{g} \to \mathtt{f}$, so $\mathcal{G} = \{A \xrightarrow{G_1} B, B \xrightarrow{G_2} B, B \xrightarrow{G_3} A\}$ where $A = \{\mathtt{x}, \mathtt{y}\}$ and $B = \{\mathtt{u}, \mathtt{v}, \mathtt{w}\}$. (Note: the vertical layout of the size-change graphs in Figure 1 is inessential, though intuitively appealing. One could simply write, for instance, $G_3 = \{\mathtt{u} \xrightarrow{\downarrow} \mathtt{x}, \mathtt{w} \xrightarrow{=} \mathtt{y}\}$.)

The termination reasoning shows that $\mathcal{G}$ satisfies the size-change condition: Every infinite multipath has either a thread that decreases u infinitely, or a thread that decreases v infinitely.

2 The Call-by-Value λ-Calculus

We first summarise some standard definitions and results.

Definition 4. *Exp is the set of all λ-expressions that can be formed by these syntax rules, where* @ *is the* application operator *(sometimes omitted). We use the* teletype *font for λ-expressions.*

```
e, P  ::= x | e @ e | λx.e
x     ::= Variable name
```

- *The set of* free variables *fv*(e) *is defined in the usual way:* $fv(\mathtt{x}) = \{\mathtt{x}\}$, $fv(\mathtt{e@e'}) = fv(\mathtt{e}) \cup fv(\mathtt{e'})$ *and* $fv(\lambda\mathtt{x.e}) = fv(\mathtt{e}) \setminus \{\mathtt{x}\}$. *A closed λ-expression* e *satisfies* $fv(\mathtt{e}) = \emptyset$.
- *A* program, *usually denoted by* P, *is any closed λ-expression.*
- *The set of* subxpressions *of a λ-expression* e *is denoted by subexp*(e).

The following is standard, e.g., [9]. Notation: $\mathsf{e}[v/\mathtt{x}]$ (β-reduction) is the result of substituting v for all free occurrences of $\mathtt{x}$ in e and renaming λ-bound variables if needed to avoid capture.

Definition 5. (Call-by-value semantics) *The* call-by-value evaluation relation *is defined by the following inference rules, with judgement form* $\mathsf{e} \Downarrow v$ *where* e *is a* λ*-expression and* $v \in$ *ValueS. ValueS (for "standard value") is the set of all abstractions* $\lambda\mathtt{x}.\mathsf{e}$.

$$\text{(ValueS)}\ \frac{}{v \Downarrow v}\ (\text{If } v \in \mathit{ValueS})$$

$$\text{(ApplyS)}\ \frac{\mathsf{e}_1 \Downarrow \lambda\mathtt{x}.\mathsf{e}_0 \qquad \mathsf{e}_2 \Downarrow v_2 \qquad \mathsf{e}_0[v_2/\mathtt{x}] \Downarrow v}{\mathsf{e}_1@\mathsf{e}_2 \Downarrow v}$$

Lemma 2. (Determinism) *If* $\mathsf{e} \Downarrow v$ *and* $\mathsf{e} \Downarrow w$ *then* $v = w$.

Lemma 3. *If* e *is closed and its* λ*-variables are all distinct, then a deduction of* $\mathsf{e} \Downarrow v$ *involves no renaming.*

3 Challenges in Termination Analysis of the λ-Calculus

The size-change termination analysis of [5] is based on several concepts including

1. Identifying nontermination as being caused by *infinite sequences of state transitions.*
2. A fixed set of *program control points.*
3. *Observable decreases* in data value sizes.
4. *Construction* of one size-change graph for each function call.
5. Finding the program's entire *control flow graph*, and the call sequences that follow it.

At first sight *all* these concepts seem to be absent from the λ-calculus, except that an application must be a call; and even then, it is not a priori clear *which* function is being called. We will show, one step at a time, that all the concepts do in fact exist in call-by-value λ-calculus evaluation.

3.1 Identifying Nontermination (Challenge 1)

We will sometimes write $\mathsf{e} \Downarrow$ to mean $\mathsf{e} \Downarrow v$ for some $v \in \mathit{ValueS}$, and write $\mathsf{e} \not\Downarrow$ to mean there is no $v \in \mathit{ValueS}$ such that $\mathsf{e} \Downarrow v$, i.e., if evaluation of e does not terminate.

A proof of $\mathsf{e} \Downarrow v$ is a finite object, and no such proof exists if the computation of e fails to terminate. In order to trace an arbitrary computation, terminating or not, we introduce the "calls" relation $\mathsf{e} \rightarrow \mathsf{e}'$. The rationale is straightforward: $\mathsf{e} \rightarrow \mathsf{e}'$ if in order to deduce $\mathsf{e} \Downarrow v$ for some value v, it is necessary first to deduce $\mathsf{e}' \Downarrow u$ for some u, i.e., some inference rule has form $\frac{\cdots\ \mathsf{e}' \Downarrow ?\ \cdots}{\mathsf{e} \Downarrow ?}$. Applying this to Definition 5 gives the following.