Martin Steffen
Gianluigi Zavattaro (Eds.)

LNCS 3535

Formal Methods for Open Object-Based Distributed Systems

7th IFIP WG 6.1 International Conference, FMOODS 2005
Athens, Greece, June 2005
Proceedings

ifip IFIP TC6

Martin Steffen Gianluigi Zavattaro (Eds.)

Formal Methods for Open Object-Based Distributed Systems

7th IFIP WG 6.1 International Conference, FMOODS 2005
Athens, Greece, June 15-17, 2005
Proceedings

Springer

Volume Editors

Martin Steffen
Christian-Albrechts-Universität zu Kiel
Institut für Informatik und Praktische Mathematik
Hermann-Rodewald-Str. 3, 24118 Kiel, Germany
E-mail: ms@informatik.uni-kiel.de

Gianluigi Zavattaro
Dipartimento di Scienze dell'Informazione
Università degli Studi di Bologna
Mura A. Zamboni, 7, 40127 Bologna, Italy
E-mail: zavattar@cs.unibo.it

Library of Congress Control Number: 2005926702

CR Subject Classification (1998): C.2.4, D.1.3, D.2, D.3, F.3, D.4

ISSN 0302-9743
ISBN-10 3-540-26181-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-26181-0 Springer Berlin Heidelberg New York

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 11494881 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3535

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

Lecture Notes in Computer Science

For information about Vols. 1–3424

please contact your bookseller or Springer

Vol. 3535: M. Steffen, G. Zavattaro (Eds.), Formal Methods for Open Object-Based Distributed Systems. X, 323 pages. 2005.

Vol. 3532: A. Gómez-Pérez, J. Euzenat (Eds.), The Semantic Web: Research and Applications. XV, 728 pages. 2005.

Vol. 3526: S.B. Cooper, B. Löwe, L. Torenvliet (Eds.), New Computational Paradigms. XVII, 574 pages. 2005.

Vol. 3525: A.E. Abdallah, C.B. Jones, J.W. Sanders (Eds.), Communicating Sequential Processes. XIV, 321 pages. 2005.

Vol. 3524: R. Barták, M. Milano (Eds.), Integration of AI and OR Techniques in Constraint Programming for Combinatorial Problems. XI, 320 pages. 2005.

Vol. 3523: J.S. Marques, N.P. de la Blanca, P. Pina (Eds.), Pattern Recognition and Image Analysis, Part II. XXVI, 733 pages. 2005.

Vol. 3522: J.S. Marques, N.P. de la Blanca, P. Pina (Eds.), Pattern Recognition and Image Analysis, Part I. XXVI, 703 pages. 2005.

Vol. 3520: O. Pastor, J. Falcão e Cunha (Eds.), Advanced Information Systems Engineering. XVI, 584 pages. 2005.

Vol. 3518: T.B. Ho, D. Cheung, H. Li (Eds.), Advances in Knowledge Discovery and Data Mining. XXI, 864 pages. 2005. (Subseries LNAI).

Vol. 3517: H.S. Baird, D.P. Lopresti (Eds.), Human Interactive Proofs. IX, 143 pages. 2005.

Vol. 3516: V.S. Sunderam, G.D. van Albada, P.M.A. Sloot, J.J. Dongarra (Eds.), Computational Science – ICCS 2005, Part III. LXIII, 1143 pages. 2005.

Vol. 3515: V.S. Sunderam, G.D. van Albada, P.M.A. Sloot, J.J. Dongarra (Eds.), Computational Science – ICCS 2005, Part II. LXIII, 1101 pages. 2005.

Vol. 3514: V.S. Sunderam, G.D. van Albada, P.M.A. Sloot, J.J. Dongarra (Eds.), Computational Science – ICCS 2005, Part I. LXIII, 1089 pages. 2005.

Vol. 3513: A. Montoyo, R. Mu\~noz, E. Métais (Eds.), Natural Language Processing and Information Systems. XII, 408 pages. 2005.

Vol. 3510: T. Braun, G. Carle, Y. Koucheryavy, V. Tsaoussidis (Eds.), Wired/Wireless Internet Communications. XIV, 366 pages. 2005.

Vol. 3509: M. Jünger, V. Kaibel (Eds.), Integer Programming and Combinatorial Optimization. XI, 484 pages. 2005.

Vol. 3508: P. Bresciani, P. Giorgini, B. Henderson-Sellers, G. Low, M. Winikoff (Eds.), Agent-Oriented Information Systems II. X, 227 pages. 2005. (Subseries LNAI).

Vol. 3507: F. Crestani, I. Ruthven (Eds.), Information Context: Nature, Impact, and Role. XIII, 253 pages. 2005.

Vol. 3505: V. Gorodetsky, J. Liu, V. A. Skormin (Eds.), Autonomous Intelligent Systems: Agents and Data Mining. XIII, 303 pages. 2005. (Subseries LNAI).

Vol. 3503: S.E. Nikoletseas (Ed.), Experimental and Efficient Algorithms. XV, 624 pages. 2005.

Vol. 3502: F. Khendek, R. Dssouli (Eds.), Testing of Communicating Systems. X, 381 pages. 2005.

Vol. 3501: B. Kégl, G. Lapalme (Eds.), Advances in Artificial Intelligence. XV, 458 pages. 2005. (Subseries LNAI).

Vol. 3500: S. Miyano, J. Mesirov, S. Kasif, S. Istrail, P. Pevzner, M. Waterman (Eds.), Research in Computational Molecular Biology. XVII, 632 pages. 2005. (Subseries LNBI).

Vol. 3499: A. Pelc, M. Raynal (Eds.), Structural Information and Communication Complexity. X, 323 pages. 2005.

Vol. 3498: J. Wang, X. Liao, Z. Yi (Eds.), Advances in Neural Networks – ISNN 2005, Part III. L, 1077 pages. 2005.

Vol. 3497: J. Wang, X. Liao, Z. Yi (Eds.), Advances in Neural Networks – ISNN 2005, Part II. L, 947 pages. 2005.

Vol. 3496: J. Wang, X. Liao, Z. Yi (Eds.), Advances in Neural Networks – ISNN 2005, Part II. L, 1055 pages. 2005.

Vol. 3495: P. Kantor, G. Muresan, F. Roberts, D.D. Zeng, F.-Y. Wang, H. Chen, R.C. Merkle (Eds.), Intelligence and Security Informatics. XVIII, 674 pages. 2005.

Vol. 3494: R. Cramer (Ed.), Advances in Cryptology – EUROCRYPT 2005. XIV, 576 pages. 2005.

Vol. 3493: N. Fuhr, M. Lalmas, S. Malik, Z. Szlávik (Eds.), Advances in XML Information Retrieval. XI, 438 pages. 2005.

Vol. 3492: P. Blache, E. Stabler, J. Busquets, R. Moot (Eds.), Logical Aspects of Computational Linguistics. X, 363 pages. 2005. (Subseries LNAI).

Vol. 3489: G.T. Heineman, I. Crnkovic, H.W. Schmidt, J.A. Stafford, C. Szyperski, K. Wallnau (Eds.), Component-Based Software Engineering. XI, 358 pages. 2005.

Vol. 3488: M.-S. Hacid, N.V. Murray, Z.W. Raś, S. Tsumoto (Eds.), Foundations of Intelligent Systems. XIII, 700 pages. 2005. (Subseries LNAI).

Vol. 3486: T. Helleseth, D. Sarwate, H.-Y. Song, K. Yang (Eds.), Sequences and Their Applications - SETA 2004. XII, 451 pages. 2005.

Vol. 3483: O. Gervasi, M.L. Gavrilova, V. Kumar, A. Laganà, H.P. Lee, Y. Mun, D. Taniar, C.J.K. Tan (Eds.), Computational Science and Its Applications – ICCSA 2005, Part IV. XXVII, 1362 pages. 2005.

Vol. 3482: O. Gervasi, M.L. Gavrilova, V. Kumar, A. Laganà, H.P. Lee, Y. Mun, D. Taniar, C.J.K. Tan (Eds.), Computational Science and Its Applications – ICCSA 2005, Part III. LXVI, 1340 pages. 2005.

Vol. 3481: O. Gervasi, M.L. Gavrilova, V. Kumar, A. Laganà, H.P. Lee, Y. Mun, D. Taniar, C.J.K. Tan (Eds.), Computational Science and Its Applications – ICCSA 2005, Part II. LXIV, 1316 pages. 2005.

Vol. 3480: O. Gervasi, M.L. Gavrilova, V. Kumar, A. Laganà, H.P. Lee, Y. Mun, D. Taniar, C.J.K. Tan (Eds.), Computational Science and Its Applications – ICCSA 2005, Part I. LXV, 1234 pages. 2005.

Vol. 3479: T. Strang, C. Linnhoff-Popien (Eds.), Location- and Context-Awareness. XII, 378 pages. 2005.

Vol. 3478: C. Jermann, A. Neumaier, D. Sam (Eds.), Global Optimization and Constraint Satisfaction. XIII, 193 pages. 2005.

Vol. 3477: P. Herrmann, V. Issarny, S. Shiu (Eds.), Trust Management. XII, 426 pages. 2005.

Vol. 3475: N. Guelfi (Ed.), Rapid Integration of Software Engineering Techniques. X, 145 pages. 2005.

Vol. 3468: H.W. Gellersen, R. Want, A. Schmidt (Eds.), Pervasive Computing. XIII, 347 pages. 2005.

Vol. 3467: J. Giesl (Ed.), Term Rewriting and Applications. XIII, 517 pages. 2005.

Vol. 3465: M. Bernardo, A. Bogliolo (Eds.), Formal Methods for Mobile Computing. VII, 271 pages. 2005.

Vol. 3464: S.A. Brueckner, G.D.M. Serugendo, A. Karageorgos, R. Nagpal (Eds.), Engineering Self-Organising Systems. XIII, 299 pages. 2005. (Subseries LNAI).

Vol. 3463: M. Dal Cin, M. Kaâniche, A. Pataricza (Eds.), Dependable Computing - EDCC 2005. XVI, 472 pages. 2005.

Vol. 3462: R. Boutaba, K.C. Almeroth, R. Puigjaner, S. Shen, J.P. Black (Eds.), NETWORKING 2005. XXX, 1483 pages. 2005.

Vol. 3461: P. Urzyczyn (Ed.), Typed Lambda Calculi and Applications. XI, 433 pages. 2005.

Vol. 3460: Ö. Babaoglu, M. Jelasity, A. Montresor, C. Fetzer, S. Leonardi, A. van Moorsel, M. van Steen (Eds.), Self-star Properties in Complex Information Systems. IX, 447 pages. 2005.

Vol. 3459: R. Kimmel, N.A. Sochen, J. Weickert (Eds.), Scale Space and PDE Methods in Computer Vision. XI, 634 pages. 2005.

Vol. 3458: P. Herrero, M.S. Pérez, V. Robles (Eds.), Scientific Applications of Grid Computing. X, 208 pages. 2005.

Vol. 3456: H. Rust, Operational Semantics for Timed Systems. XII, 223 pages. 2005.

Vol. 3455: H. Treharne, S. King, M. Henson, S. Schneider (Eds.), ZB 2005: Formal Specification and Development in Z and B. XV, 493 pages. 2005.

Vol. 3454: J.-M. Jacquet, G.P. Picco (Eds.), Coordination Models and Languages. X, 299 pages. 2005.

Vol. 3453: L. Zhou, B.C. Ooi, X. Meng (Eds.), Database Systems for Advanced Applications. XXVII, 929 pages. 2005.

Vol. 3452: F. Baader, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning. XI, 562 pages. 2005. (Subseries LNAI).

Vol. 3450: D. Hutter, M. Ullmann (Eds.), Security in Pervasive Computing. XI, 239 pages. 2005.

Vol. 3449: F. Rothlauf, J. Branke, S. Cagnoni, D.W. Corne, R. Drechsler, Y. Jin, P. Machado, E. Marchiori, J. Romero, G.D. Smith, G. Squillero (Eds.), Applications of Evolutionary Computing. XX, 631 pages. 2005.

Vol. 3448: G.R. Raidl, J. Gottlieb (Eds.), Evolutionary Computation in Combinatorial Optimization. XI, 271 pages. 2005.

Vol. 3447: M. Keijzer, A. Tettamanzi, P. Collet, J.v. Hemert, M. Tomassini (Eds.), Genetic Programming. XIII, 382 pages. 2005.

Vol. 3444: M. Sagiv (Ed.), Programming Languages and Systems. XIII, 439 pages. 2005.

Vol. 3443: R. Bodik (Ed.), Compiler Construction. XI, 305 pages. 2005.

Vol. 3442: M. Cerioli (Ed.), Fundamental Approaches to Software Engineering. XIII, 373 pages. 2005.

Vol. 3441: V. Sassone (Ed.), Foundations of Software Science and Computational Structures. XVIII, 521 pages. 2005.

Vol. 3440: N. Halbwachs, L.D. Zuck (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. XVII, 588 pages. 2005.

Vol. 3439: R.H. Deng, F. Bao, H. Pang, J. Zhou (Eds.), Information Security Practice and Experience. XII, 424 pages. 2005.

Vol. 3438: H. Christiansen, P.R. Skadhauge, J. Villadsen (Eds.), Constraint Solving and Language Processing. VIII, 205 pages. 2005. (Subseries LNAI).

Vol. 3437: T. Gschwind, C. Mascolo (Eds.), Software Engineering and Middleware. X, 245 pages. 2005.

Vol. 3436: B. Bouyssounouse, J. Sifakis (Eds.), Embedded Systems Design. XV, 492 pages. 2005.

Vol. 3434: L. Brun, M. Vento (Eds.), Graph-Based Representations in Pattern Recognition. XII, 384 pages. 2005.

Vol. 3433: S. Bhalla (Ed.), Databases in Networked Information Systems. VII, 319 pages. 2005.

Vol. 3432: M. Beigl, P. Lukowicz (Eds.), Systems Aspects in Organic and Pervasive Computing - ARCS 2005. X, 265 pages. 2005.

Vol. 3431: C. Dovrolis (Ed.), Passive and Active Network Measurement. XII, 374 pages. 2005.

Vol. 3430: S. Tsumoto, T. Yamaguchi, M. Numao, H. Motoda (Eds.), Active Mining. XII, 349 pages. 2005. (Subseries LNAI).

Vol. 3429: E. Andres, G. Damiand, P. Lienhardt (Eds.), Discrete Geometry for Computer Imagery. X, 428 pages. 2005.

Vol. 3428: Y.-J. Kwon, A. Bouju, C. Claramunt (Eds.), Web and Wireless Geographical Information Systems. XII, 255 pages. 2005.

Vol. 3427: G. Kotsis, O. Spaniol (Eds.), Wireless Systems and Mobility in Next Generation Internet. VIII, 249 pages. 2005.

¥641.92元

Preface

This volume contains the proceedings of FMOODS 2005, the 7th IFIP WG 6.1 International Conference on *Formal Methods for Open Object-Based Distributed Systems*. The conference was held in Athens, Greece on June 15–17, 2005. The event was the seventh meeting of this conference series, which is held roughly every year and a half, with the earlier events held respectively in Paris, Canterbury, Florence, Stanford, Twente, and Paris.

The goal of the FMOODS series of conferences is to bring together researchers whose work encompasses three important and related fields:

- formal methods;
- distributed systems;
- object-based technology.

Such a convergence is representative of recent advances in the field of distributed systems, and provides links between several scientific and technological communities, as represented by the conferences FORTE, CONCUR, and ECOOP.

The objective of FMOODS is to provide an integrated forum for the presentation of research in the above-mentioned fields, and the exchange of ideas and experiences in the topics concerned with the formal methods support for open object-based distributed systems. For the call for papers, aspects of interest included, but were not limited to: formal models; formal techniques for specification, design, or analysis; verification, testing, and validation; component-based design; formal aspects of service-oriented computing; semantics and type systems for programming, coordination, or modelling languages; behavioral typing; multiple viewpoint modelling and consistency between different models; transformations of models; integration of quality-of-service requirements into formal models; formal models for security; formal approaches to distributed component frameworks; and applications and experience, carefully described. Work on these aspects of (official and de facto) standard notation and languages for service oriented design, e.g. web services orchestration languages, was explicitly welcome.

In total 49 abstracts and 42 papers were submitted to this year's conference, covering the full range of topics listed above. Out of the submissions, 19 research papers were selected by the Program Committee for presentation. We would like to express our deepest appreciation to the authors of all submitted papers and to the Program Committee members and external reviewers who did an outstanding job in selecting the best papers for presentation.

For the second time, the FMOODS conference was held as a joint event, this time in federation with the 5th IFIP WG 6.1 International Conference on *Distributed Applications and Interoperable Systems* (DAIS 2005). The co-location of the FMOODS and DAIS conferences provided an excellent opportunity to the participants for a wide and comprehensive exchange of ideas within the domain of distributed systems and applications. Both FMOODS and DAIS address this

domain, the former with its emphasis on formal approaches the latter on practical solutions. Their combination in a single event ensured that both theoretical foundations and practical issues were presented and discussed.

Special thanks to Lazaros Merakos, for acting as the General Chair of the joint conferences DAIS and FMOODS 2005; his support made this event happen. We would also like to thank Gordon Blair, Rocco de Nicola, and Andreas Reuter for agreeing to present invited talks at the conference.

We thank Costas Polychronopoulos, for acting as Local Arrangements Chair, and John Derrick for his work as Publicity Chair. We would also like to thank the FMOODS Steering Committee (John Derrick, Roberto Gorrieri, Guy Leduc, and Elie Najm) for their advice. Thanks also to Roberto Lucchi for his valuable help in managing the submission server.

June 2005

Martin Steffen
Gianluigi Zavattaro

Organization

General Chair	Lazaros Merakos (University of Athens, Greece)
Program Chairs	Martin Steffen (University of Kiel, Germany) Gianluigi Zavattaro (University of Bologna, Italy)
Local Arrangements	Costas Polychronopoulos (University of Athens, Greece)
Publicity Chair	John Derrick (University of Kent, UK)

Steering Committee

John Derrick
Roberto Gorrieri
Guy Leduc
Elie Najm

Program Committee

Wil van der Aalst (Netherlands)
Lynne Blair (UK)
Frank van Breugel (Canada)
Michele Bugliesi (Italy)
John Derrick (UK)
Sophia Drossopoulou (UK)
Alessandro Fantechi (Italy)
Kokichi Futatsugi (Japan)
Andy Gordon (UK)
Roberto Gorrieri (Italy)
Jan Jürjens (Germany)
Cosimo Laneve (Italy)
Luigi Logrippo (Canada)
Elie Najm (France)
Uwe Nestmann (Switzerland)
Ernesto Pimentel (Spain)
Erik Poll (Netherlands)
Andreas Prinz (Norway)
Arend Rensink (Netherlands)
Bernhard Rumpe (Germany)
Martin Steffen (Germany)
Perdita Stevens (UK)
Carolyn Talcott (USA)
Vasco Thudichum Vasconcelos (Portugal)
Nalini Venkatasubramanian (USA)
Heike Wehrheim (Germany)
Gianluigi Zavattaro (Italy)

Referees

Kamel Adi
Andreas Bauer
Johannes Borgström
Alex Buckley
Luis Caires
Ugo Dal Lago
Ferruccio Damiani
Grit Denker
Susan Eisenbach
Harald Fecher
Rachele Fuzzati
Borislav Gajanovic
Vladimir Gapeyev
Simon Gay
Sebastian Gutierrez-Nolasco
William Heaven
Marcel Kyas
Roberto Lucchi
Francisco Martins
Franco Mazzanti
Masaki Nakamura
Kazuhiro Ogata
Martijn Oostdijk
Liviu Pene
G. Michele Pinna
Andreas Schäfer
Takahiro Seino
Peter Sewell
Matthew Smith
Christoph Sprenger
Emilio Tuosto
Björn Victor
Martijn Warnier
Lucian Wischik

Table of Contents

Analysis of Java Programs

Web Services

Specification and Verification

Pattern Matching over a Dynamic Network of Tuple Spaces

Rocco De Nicola[1], Daniele Gorla[2,*], and Rosario Pugliese[1]

[1] Dipartimento di Sistemi e Informatica, Università di Firenze
{denicola,pugliese}@dsi.unifi.it
[2] Dipartimento di Informatica, Università di Roma "La Sapienza"
gorla@di.uniroma1.it

Abstract. In this paper, we present recent work carried on μKLAIM, a core calculus that retains most of the features of KLAIM: explicit process distribution, remote operations, process mobility and asynchronous communication via distributed tuple spaces. Communication in μKLAIM is based on a simple form of pattern matching that enables withdrawal from shared data spaces of matching tuples and binds the matched variables within the continuation process. Pattern matching is orthogonal to the underlying computational paradigm of μKLAIM, but affects its expressive power. After presenting the basic pattern matching mechanism, inherited from KLAIM, we discuss a number of variants that are easy to implement and test, by means of simple examples, the expressive power of the resulting variants of the language.

1 Introduction

In the last decade, programming computational infrastructures available globally for offering uniform services has become one of the main issues in Computer Science. The challenges come from the necessity of dealing at once with issues like communication, co-operation, mobility, resource usage, security, privacy, failures, etc. in a setting where demands and guarantees can be very different for the many different components. KLAIM (*Kernel Language for Agents Interaction and Mobility*, [5]) is a tentative response to the call for innovative theories, computational paradigms, linguistic mechanisms and implementation techniques for the design, realization, deployment and management of global computational environments and their application.

KLAIM is an experimental language specifically designed to program distributed systems made up of several mobile components interacting through multiple distributed tuple spaces. Its communication model builds over, and extends, LINDA's notion of generative communication through a shared tuple space [11]. The LINDA model was originally proposed for parallel programming on isolated machines; multiple, possibly distributed, tuple spaces have been advocated later [12] to improve modularity, scalability and performance, and fit well in a global computing scenario.

* Most of the work presented in this paper was carried on while the second author was a PhD student at the University of Florence.

M. Steffen and G. Zavattaro (Eds.): FMOODS 2005, LNCS 3535, pp. 1–14, 2005.

Table 1. μKLAIM Syntax

NETS $N ::= \mathbf{0} \mid l :: C \mid N_1 \parallel N_2 \mid (\nu l)N$	COMPONENTS $C ::= \langle t \rangle \mid P \mid C_1 \mid C_2$
TUPLES $t ::= u \mid t_1, t_2$	TEMPLATES $T ::= u \mid !x \mid T_1, T_2$
ACTIONS $a ::= \mathbf{in}(T)@u \mid \mathbf{read}(T)@u \mid \mathbf{out}(t)@u \mid \mathbf{eval}(P)@u \mid \mathbf{new}(l)$	
PROCESSES $P ::= \mathbf{nil} \mid a.P \mid P_1 \mid P_2 \mid *P$	

KLAIM has proved to be suitable for programming a wide range of distributed applications with agents and code mobility [5, 6] and it has originated an actual programming language, X-KLAIM [1], that has been implemented by exploiting Java [2].

The main drawback of KLAIM is that it is not an actual programming language, nor a process calculus. The main aim of some our recent works (grouped together in [13]) has been the definition of a process calculus derived from KLAIM that retains all its distinctive features and expressive power, and develop over it the type theoretic and semantical foundations of the language. The resulting calculus has been called μKLAIM and, in [8], we have proved that it can reasonably encode KLAIM.

In this paper, we first describe μKLAIM (Section 2). Then, in Section 3, we present some recent enhancements of the basic formalism to deal with some low-level features, namely inter-node connections and failures. In Section 4, we argue on alternative forms of pattern matching for retrieving tuples. So far, KLAIM and its variants have used LINDA's original pattern matching, because of its simplicity. Nevertheless, other variants could be adopted without compromising language implementability, actually enhancing the overall expressive power. A novel contribution of this paper is the informal examination of this topic. Section 5 concludes the paper.

2 The Calculus μKLAIM

2.1 Syntax

The syntax of μKLAIM is reported in Table 1. A countable set $\mathcal{L}$ of *names* $l, l', \ldots, u, \ldots, x, y, \ldots$ is assumed. Names provide the abstract counterpart of the set of *communicable* objects and can be used as localities and variables: we do not distinguish between these kinds of objects. Notationally, we prefer letters $l, l', \ldots$ when we want to stress the use of a name as a locality and $x, y, \ldots$ when we want to stress the use of a name as a variable. We will use u for basic variables and localities.

Nets are finite collections of nodes where processes and tuple spaces can be allocated. A *node* is a pair $l :: C$, where locality l is the address of the node and C is the parallel component located at l. *Components* can be processes or (located) tuples. *Located tuples*, $\langle t \rangle$, are inactive components representing tuples in a tuple space (TS, for

Table 2. The Pattern Matching Function

$match(l; l) = \epsilon$	$\dfrac{match(T_1; t_1) = \sigma_1 \quad match(T_2; t_2) = \sigma_2}{match(T_1, T_2; t_1, t_2) = \sigma_1 \circ \sigma_2}$
$match(!x; l) = [^l/_x]$	

short) that have been inserted either in the initial configuration or along a computation by executing an action **out**. The TS located at l results from the parallel composition of all located tuples residing at l. In $(\nu l)N$, name l is private to N; the intended effect is that, if one considers the term $N_1 \parallel (\nu l)N_2$, then locality l of N_2 cannot be referred from within N_1.

Tuples are sequences of names. *Templates* are patterns used to select tuples in a TS. They are sequences of names and formal fields; the latter ones are written $!\,x$ and are used to bind variables to names.

Processes are the μKLAIM active computational units. They are built up from the inert process **nil** and from five basic operations, called *actions*, by using action prefixing, parallel composition and replication. The informal semantics of process actions is as follows. Action $\mathbf{in}(T)@u$ looks for a matching tuple $\langle t \rangle$ in the TS located at u; intuitively, a template matches against a tuple if both have the same number of fields and corresponding fields match, i.e. they are the same name, or one is a formal while the other one is a name. If $\langle t \rangle$ is found, it is removed from the TS, the formal fields of T are replaced in the continuation process with the corresponding names of t and the operation terminates. If no matching tuple is found, the operation is suspended until one is available. Action $\mathbf{read}(T)@u$ is similar but it leaves the selected tuple in u's TS. Action $\mathbf{out}(t)@u$ adds the tuple t to the TS located at u. Action $\mathbf{eval}(P)@u$ sends process P for execution at u. Action $\mathbf{new}(l)$ creates a new node in the net at the reserved address l. Notice that **new** is the only action not indexed with an address because it always acts locally; all the other actions explicitly indicate the (possibly remote) locality where they will take place.

Names occurring in terms can be bound by action prefixes or by restriction. More precisely, in processes $\mathbf{in}(T)@u.P$ and $\mathbf{read}(T)@u.P$ the prefixes bind the names in the formal fields of T within P; in process $\mathbf{new}(l).P$, the prefix binds l in P; in $(\nu l)N$, the restriction binds l in N. A name that is not bound is called *free*. The sets $bn(\cdot)$ and $fn(\cdot)$ (of bound and free names, resp., of term $\cdot$) are defined accordingly, and so is *alpha-conversion*. In the sequel, we shall assume that bound names in terms are all distinct and different from the free ones (by possibly applying alpha-conversion, this requirement can always be satisfied).

2.2 Operational Semantics

μKLAIM operational semantics is given in terms of a structural congruence and a reduction relation. The *structural congruence*, $\equiv$, identifies nets which intuitively represent the same net. It is inspired to π-calculus' structural congruence (see, e.g., [16]) and states that '$\parallel$' is a monoidal operator with $\mathbf{0}$ as identity, that **nil** is the identity for '|', that alpha-equivalent nets do coincide, and that the order of restrictions in a net is irrelevant.

Table 3. μKLAIM Reduction Relation

$$\text{(R-Out)}\quad \frac{}{l :: \mathbf{out}(t)@l'.P \parallel l' :: \mathbf{nil} \longmapsto l :: P \parallel l' :: \langle t \rangle} \qquad \text{(R-New)}\quad \frac{}{l :: \mathbf{new}(l').P \longmapsto (\nu l')(l :: P \parallel l' :: \mathbf{nil})}$$

$$\text{(R-Eval)}\quad \frac{}{l :: \mathbf{eval}(P_2)@l'.P_1 \parallel l' :: \mathbf{nil} \longmapsto l :: P_1 \parallel l' :: P_2} \qquad \text{(R-Res)}\quad \frac{N \longmapsto N'}{(\nu l)N \longmapsto (\nu l)N'}$$

$$\text{(R-In)}\quad \frac{match(T;t) = \sigma}{l :: \mathbf{in}(T)@l'.P \parallel l' :: \langle t \rangle \longmapsto l :: P\sigma \parallel l' :: \mathbf{nil}} \qquad \text{(R-Par)}\quad \frac{N_1 \longmapsto N_1'}{N_1 \parallel N_2 \longmapsto N_1' \parallel N_2}$$

$$\text{(R-Read)}\quad \frac{match(T;t) = \sigma}{l :: \mathbf{read}(T)@l'.P \parallel l' :: \langle t \rangle \longmapsto l :: P\sigma \parallel l' :: \langle t \rangle} \qquad \text{(R-Struct)}\quad \frac{N \equiv N_1 \quad N_1 \longmapsto N_2 \quad N_2 \equiv N'}{N \longmapsto N'}$$

Moreover, the following laws are crucial to our setting:

$$\text{(Clone)}\quad l :: C_1 | C_2 \equiv l :: C_1 \parallel l :: C_2$$

$$\text{(Repl)}\quad l :: *P \equiv l :: P \mid *P$$

$$\text{(RepNil)}\quad l :: *\mathbf{nil} \equiv l :: \mathbf{nil}$$

$$\text{(Ext)}\quad N_1 \parallel (\nu l)N_2 \equiv (\nu l)(N_1 \parallel N_2) \quad \text{if } l \notin fn(N_1)$$

Law (Clone) turns a parallel between co-located components into a parallel between nodes (by relying on this law, commutativity and associativity of '|' follows). Law (Repl) unfolds a replicated process; however, when the replicated process is **nil**, the unfolding is useless (see rule (RepNil)). Finally, law (Ext) is the standard π-calculus' rule for scope extension; it states that the scope of a restricted name can be extended, provided that no free name is captured.

The reduction relation is given in Table 3. It relies on the *pattern matching* function $match(\_;\_)$ that verifies the compliance of a tuple w.r.t. a template and associates values to variables bound in the template. Intuitively, a tuple matches a template if they have the same number of fields, and corresponding fields match. Formally, function *match* is defined in Table 2 where we let 'ϵ' be the empty substitution and '$\circ$' denote substitutions composition. Here, a substitution σ is a mapping of names for names; $P\sigma$ denotes the (capture avoiding) application of σ to P.

The operational rules of μKLAIM can be briefly motivated as follows. Rule (R-Out) states that execution of an output sends the tuple argument of the action to the target node. However, this is possible only if the target node does exist in the net. Rule (R-Eval) is similar, but deals with process spawning. Rules (R-In) and (R-Read) require existence of a matching datum in the target node. The tuple is then used to replace the free occurrences of the variables bound by the template in the continuation of the process performing the actions. With action **in** the matched datum is consumed while

with action **read** it is not. Rule (R-NEW) states that action **new**(l') creates a new node at a reserved address l'. Rules (R-PAR), (R-RES) and (R-STRUCT) are standard.

μKLAIM adopts a LINDA-like [11] communication mechanism: data are anonymous and associatively accessed via pattern matching, and communication is asynchronous. Indeed, even if there exist action prefixes for placing data to (possibly remote) nodes, no synchronization takes place between (sending and receiving) processes, because their interactions are mediated by nodes, that act as data repositories.

2.3 Observational Semantics

We now present a preorder on μKLAIM nets yielding sensible semantic theories. We follow the approach put forward in [10] and use *may testing* equivalence. Intuitively, two nets are may testing equivalent if they cannot be distinguished by any external observer taking note of the data offered by the observed net. More precisely, an *observer* O is a net containing a node whose address is a reserved locality name `test`. A computation reports *success* if, along its execution, a datum at node `test` appears; this is written $\stackrel{OK}{\Longrightarrow}$.

Definition 1 (May Testing Equivalence). May testing, $\sqsubseteq$, *is the least equivalence on* μKLAIM *nets such that, for every* $N \sqsubseteq M$, *it holds that* $N \parallel O \stackrel{OK}{\Longrightarrow}$ *if and only if* $M \parallel O \stackrel{OK}{\Longrightarrow}$, *for any observer* O.

The problem underneath the definition of may testing we have just presented is the universal quantification over observers. This makes it hard to prove equivalences in practice. In [13], we have developed an alternative characterisations of $\simeq$ as a trace-based equivalence and a co-inductive proof technique as a bisimulation-based equivalence. However, these definitions have been omitted from this paper: here, it sufficies to have a sensible notion of equivalence to equate nets.

3 Node Connections and Failures

In this section we present two enhancements of the basic framework presented so far. Such enhancements allow us to better model some global computing phenomena.

3.1 Modelling Connections

In [7], we developed the behavioural theory of a language derived from μKLAIM by introducing explicit inter-node connections and process actions to dynamically change them. The syntax of the resulting calculus, that is called TKLAIM (*topological* KLAIM), can be obtained by adding the following productions to those in Table 1:

$$N ::= \cdots \mid \{l_1 \rightarrow l_2\} \qquad a ::= \cdots \mid \mathbf{conn}(u) \mid \mathbf{disc}(u)$$

A *connection* (or *link*) is a pair of node addresses $\{l_1 \rightarrow l_2\}$ stating that the nodes at addresses l_1 and l_2 are directly linked. Actions $\mathbf{conn}(l_2)$ and $\mathbf{disc}(l_2)$ aim at changing