Rudolf Freund
Gheorghe Păun
Grzegorz Rozenberg
Arto Salomaa (Eds.)

LNCS 3850

Membrane Computing

6th International Workshop, WMC 2005
Vienna, Austria, July 2005
Revised Selected and Invited Papers

Springer

Rudolf Freund Gheorghe Păun
Grzegorz Rozenberg Arto Salomaa (Eds.)

Membrane Computing

6th International Workshop, WMC 2005
Vienna, Austria, July 18-21, 2005
Revised Selected and Invited Papers

Springer

Volume Editors

Rudolf Freund
Vienna University of Technology
Faculty of Informatics
Favoritenstr. 9–11, 1040 Vienna, Austria
E-mail: rudi@emcc.at

Gheorghe Păun
Institute of Mathematics of the Romanian Academy
P.O. Box 1-764, 014700 Bucureşti, Romania
and
Sevilla University, Dept. of Computer Science and AI
Research Group on Natural Computing
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
E-mail: gpaun@us.es

Grzegorz Rozenberg
Leiden University
Leiden Center of Advanced Computer Science (LIACS)
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
E-mail: rozenber@liacs.nl

Arto Salomaa
Turku Centre for Computer Science (TUCS)
Leminkäisenkatu 14, 20520 Turku, Finland
E-mail: asalomaa@cs.utu.fi

Library of Congress Control Number: 2005937334

CR Subject Classification (1998): F.1, F.4, I.6, J.3

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743
ISBN-10 3-540-30948-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-30948-2 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11603047 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3850

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

Editorial Board

Author Index

Lecture Notes in Computer Science

For information about Vols. 1–3742

please contact your bookseller or Springer

Vol. 3850: R. Freund, G. Păun, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing. IX, 371 pages. 2006.

Vol. 3838: A. Middeldorp, V. van Oostrom, F. van Raamsdonk, R. de Vrijer (Eds.), Processes, Terms and Cycles: Steps on the Road to Infinity. XVIII, 639 pages. 2005.

Vol. 3837: K. Cho, P. Jacquet (Eds.), Technologies for Advanced Heterogeneous Networks. IX, 307 pages. 2005.

Vol. 3835: G. Sutcliffe, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning. XIV, 744 pages. 2005. (Sublibrary LNAI).

Vol. 3833: K.-J. Li, C. Vangenot (Eds.), Web and Wireless Geographical Information Systems. XI, 309 pages. 2005.

Vol. 3829: P. Pettersson, W. Yi (Eds.), Formal Modeling and Analysis of Timed Systems. IX, 305 pages. 2005.

Vol. 3828: X. Deng, Y. Ye (Eds.), Internet and Network Economics. XVII, 1106 pages. 2005.

Vol. 3827: X. Deng, D. Du (Eds.), Algorithms and Computation. XX, 1190 pages. 2005.

Vol. 3826: B. Benatallah, F. Casati, P. Traverso (Eds.), Service-Oriented Computing - ICSOC 2005. XVIII, 597 pages. 2005.

Vol. 3824: L.T. Yang, M. Amamiya, Z. Liu, M. Guo, F.J. Rammig (Eds.), Embedded and Ubiquitous Computing – EUC 2005. XXIII, 1204 pages. 2005.

Vol. 3823: T. Enokido, L. Yan, B. Xiao, D. Kim, Y. Dai, L.T. Yang (Eds.), Embedded and Ubiquitous Computing – EUC 2005 Workshops. XXXII, 1317 pages. 2005.

Vol. 3822: D. Feng, D. Lin, M. Yung (Eds.), Information Security and Cryptology. XII, 420 pages. 2005.

Vol. 3821: R. Ramanujam, S. Sen (Eds.), FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science. XIV, 566 pages. 2005.

Vol. 3820: L.T. Yang, X. Zhou, W. Zhao, Z. Wu, Y. Zhu, M. Lin (Eds.), Embedded Software and Systems. XXVIII, 779 pages. 2005.

Vol. 3819: P. Van Hentenryck (Ed.), Practical Aspects of Declarative Languages. X, 231 pages. 2006.

Vol. 3818: S. Grumbach, L. Sui, V. Vianu (Eds.), Advances in Computer Science – ASIAN 2005. XIII, 294 pages. 2005.

Vol. 3815: E.A. Fox, E.J. Neuhold, P. Premsmit, V. Wuwongse (Eds.), Digital Libraries: Implementing Strategies and Sharing Experiences. XVII, 529 pages. 2005.

Vol. 3814: M. Maybury, O. Stock, W. Wahlster (Eds.), Intelligent Technologies for Interactive Entertainment. XV, 342 pages. 2005. (Sublibrary LNAI).

Vol. 3810: Y.G. Desmedt, H. Wang, Y. Mu, Y. Li (Eds.), Cryptology and Network Security. XI, 349 pages. 2005.

Vol. 3809: S. Zhang, R. Jarvis (Eds.), AI 2005: Advances in Artificial Intelligence. XXVII, 1344 pages. 2005. (Sublibrary LNAI).

Vol. 3808: C. Bento, A. Cardoso, G. Dias (Eds.), Progress in Artificial Intelligence. XVIII, 704 pages. 2005. (Sublibrary LNAI).

Vol. 3807: M. Dean, Y. Guo, W. Jun, R. Kaschek, S. Krishnaswamy, Z. Pan, Q.Z. Sheng (Eds.), Web Information Systems Engineering – WISE 2005 Workshops. XV, 275 pages. 2005.

Vol. 3806: A.H. H. Ngu, M. Kitsuregawa, E.J. Neuhold, J.-Y. Chung, Q.Z. Sheng (Eds.), Web Information Systems Engineering – WISE 2005. XXI, 771 pages. 2005.

Vol. 3805: G. Subsol (Ed.), Virtual Storytelling. XII, 289 pages. 2005.

Vol. 3804: G. Bebis, R. Boyle, D. Koracin, B. Parvin (Eds.), Advances in Visual Computing. XX, 755 pages. 2005.

Vol. 3803: S. Jajodia, C. Mazumdar (Eds.), Information Systems Security. XI, 342 pages. 2005.

Vol. 3802: Y. Hao, J. Liu, Y.-P. Wang, Y.-m. Cheung, H. Yin, L. Jiao, J. Ma, Y.-C. Jiao (Eds.), Computational Intelligence and Security, Part II. XLII, 1166 pages. 2005. (Sublibrary LNAI).

Vol. 3801: Y. Hao, J. Liu, Y.-P. Wang, Y.-m. Cheung, H. Yin, L. Jiao, J. Ma, Y.-C. Jiao (Eds.), Computational Intelligence and Security, Part I. XLI, 1122 pages. 2005. (Sublibrary LNAI).

Vol. 3799: M. A. Rodríguez, I.F. Cruz, S. Levashkin, M.J. Egenhofer (Eds.), GeoSpatial Semantics. X, 259 pages. 2005.

Vol. 3798: A. Dearle, S. Eisenbach (Eds.), Component Deployment. X, 197 pages. 2005.

Vol. 3797: S. Maitra, C. E. V. Madhavan, R. Venkatesan (Eds.), Progress in Cryptology - INDOCRYPT 2005. XIV, 417 pages. 2005.

Vol. 3796: N.P. Smart (Ed.), Cryptography and Coding. XI, 461 pages. 2005.

Vol. 3795: H. Zhuge, G.C. Fox (Eds.), Grid and Cooperative Computing - GCC 2005. XXI, 1203 pages. 2005.

Vol. 3794: X. Jia, J. Wu, Y. He (Eds.), Mobile Ad-hoc and Sensor Networks. XX, 1136 pages. 2005.

Vol. 3793: T. Conte, N. Navarro, W.-m.W. Hwu, M. Valero, T. Ungerer (Eds.), High Performance Embedded Architectures and Compilers. XIII, 317 pages. 2005.

Vol. 3792: I. Richardson, P. Abrahamsson, R. Messnarz (Eds.), Software Process Improvement. VIII, 215 pages. 2005.

Vol. 3791: A. Adi, S. Stoutenburg, S. Tabet (Eds.), Rules and Rule Markup Languages for the Semantic Web. X, 225 pages. 2005.

Vol. 3790: G. Alonso (Ed.), Middleware 2005. XIII, 443 pages. 2005.

Vol. 3789: A. Gelbukh, Á. de Albornoz, H. Terashima-Marín (Eds.), MICAI 2005: Advances in Artificial Intelligence. XXVI, 1198 pages. 2005. (Sublibrary LNAI).

Vol. 3788: B. Roy (Ed.), Advances in Cryptology - ASIACRYPT 2005. XIV, 703 pages. 2005.

Vol. 3785: K.-K. Lau, R. Banach (Eds.), Formal Methods and Software Engineering. XIV, 496 pages. 2005.

Vol. 3784: J. Tao, T. Tan, R.W. Picard (Eds.), Affective Computing and Intelligent Interaction. XIX, 1008 pages. 2005.

Vol. 3783: S. Qing, W. Mao, J. Lopez, G. Wang (Eds.), Information and Communications Security. XIV, 492 pages. 2005.

Vol. 3781: S.Z. Li, Z. Sun, T. Tan, S. Pankanti, G. Chollet, D. Zhang (Eds.), Advances in Biometric Person Authentication. XI, 250 pages. 2005.

Vol. 3780: K. Yi (Ed.), Programming Languages and Systems. XI, 435 pages. 2005.

Vol. 3779: H. Jin, D. Reed, W. Jiang (Eds.), Network and Parallel Computing. XV, 513 pages. 2005.

Vol. 3778: C. Atkinson, C. Bunse, H.-G. Gross, C. Peper (Eds.), Component-Based Software Development for Embedded Systems. VIII, 345 pages. 2005.

Vol. 3777: O.B. Lupanov, O.M. Kasim-Zade, A.V. Chaskin, K. Steinhöfel (Eds.), Stochastic Algorithms: Foundations and Applications. VIII, 239 pages. 2005.

Vol. 3776: S.K. Pal, S. Bandyopadhyay, S. Biswas (Eds.), Pattern Recognition and Machine Intelligence. XXIV, 808 pages. 2005.

Vol. 3775: J. Schönwälder, J. Serrat (Eds.), Ambient Networks. XIII, 281 pages. 2005.

Vol. 3774: G. Bierman, C. Koch (Eds.), Database Programming Languages. X, 295 pages. 2005.

Vol. 3773: A. Sanfeliu, M.L. Cortés (Eds.), Progress in Pattern Recognition, Image Analysis and Applications. XX, 1094 pages. 2005.

Vol. 3772: M. Consens, G. Navarro (Eds.), String Processing and Information Retrieval. XIV, 406 pages. 2005.

Vol. 3771: J.M.T. Romijn, G.P. Smith, J. van de Pol (Eds.), Integrated Formal Methods. XI, 407 pages. 2005.

Vol. 3770: J. Akoka, S.W. Liddle, I.-Y. Song, M. Bertolotto, I. Comyn-Wattiau, W.-J. van den Heuvel, M. Kolp, J. Trujillo, C. Kop, H.C. Mayr (Eds.), Perspectives in Conceptual Modeling. XXII, 476 pages. 2005.

Vol. 3769: D.A. Bader, M. Parashar, V. Sridhar, V.K. Prasanna (Eds.), High Performance Computing – HiPC 2005. XXVIII, 550 pages. 2005.

Vol. 3768: Y.-S. Ho, H.J. Kim (Eds.), Advances in Multimedia Information Processing - PCM 2005, Part II. XXVIII, 1088 pages. 2005.

Vol. 3767: Y.-S. Ho, H.J. Kim (Eds.), Advances in Multimedia Information Processing - PCM 2005, Part I. XXVIII, 1022 pages. 2005.

Vol. 3766: N. Sebe, M.S. Lew, T.S. Huang (Eds.), Computer Vision in Human-Computer Interaction. X, 231 pages. 2005.

Vol. 3765: Y. Liu, T. Jiang, C. Zhang (Eds.), Computer Vision for Biomedical Image Applications. X, 563 pages. 2005.

Vol. 3764: S. Tixeuil, T. Herman (Eds.), Self-Stabilizing Systems. VIII, 229 pages. 2005.

Vol. 3762: R. Meersman, Z. Tari, P. Herrero (Eds.), On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops. XXXI, 1228 pages. 2005.

Vol. 3761: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, Part II. XXVII, 653 pages. 2005.

Vol. 3760: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, Part I. XXVII, 921 pages. 2005.

Vol. 3759: G. Chen, Y. Pan, M. Guo, J. Lu (Eds.), Parallel and Distributed Processing and Applications - ISPA 2005 Workshops. XIII, 669 pages. 2005.

Vol. 3758: Y. Pan, D.-x. Chen, M. Guo, J. Cao, J.J. Dongarra (Eds.), Parallel and Distributed Processing and Applications. XXIII, 1162 pages. 2005.

Vol. 3757: A. Rangarajan, B. Vemuri, A.L. Yuille (Eds.), Energy Minimization Methods in Computer Vision and Pattern Recognition. XII, 666 pages. 2005.

Vol. 3756: J. Cao, W. Nejdl, M. Xu (Eds.), Advanced Parallel Processing Technologies. XIV, 526 pages. 2005.

Vol. 3754: J. Dalmau Royo, G. Hasegawa (Eds.), Management of Multimedia Networks and Services. XII, 384 pages. 2005.

Vol. 3753: O.F. Olsen, L.M.J. Florack, A. Kuijper (Eds.), Deep Structure, Singularities, and Computer Vision. X, 259 pages. 2005.

Vol. 3752: N. Paragios, O. Faugeras, T. Chan, C. Schnörr (Eds.), Variational, Geometric, and Level Set Methods in Computer Vision. XI, 369 pages. 2005.

Vol. 3751: T. Magedanz, E.R.M. Madeira, P. Dini (Eds.), Operations and Management in IP-Based Networks. X, 213 pages. 2005.

Vol. 3750: J.S. Duncan, G. Gerig (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2005, Part II. XL, 1018 pages. 2005.

Vol. 3749: J.S. Duncan, G. Gerig (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2005, Part I. XXXIX, 942 pages. 2005.

Vol. 3748: A. Hartman, D. Kreische (Eds.), Model Driven Architecture – Foundations and Applications. IX, 349 pages. 2005.

Vol. 3747: C.A. Maziero, J.G. Silva, A.M.S. Andrade, F.M.d. Assis Silva (Eds.), Dependable Computing. XV, 267 pages. 2005.

Vol. 3746: P. Bozanis, E.N. Houstis (Eds.), Advances in Informatics. XIX, 879 pages. 2005.

Vol. 3745: J.L. Oliveira, V. Maojo, F. Martín-Sánchez, A.S. Pereira (Eds.), Biological and Medical Data Analysis. XII, 422 pages. 2005. (Sublibrary LNBI).

Vol. 3744: T. Magedanz, A. Karmouch, S. Pierre, I.S. Venieris (Eds.), Mobility Aware Technologies and Applications. XIV, 418 pages. 2005.

Preface

The present volume is based on papers presented at the 6th Workshop on Membrane Computing, WMC6, which took place in Vienna, Austria, in the period July 18–21, 2005. The first three workshops were organized in Curtea de Argeş, Romania – they took place in August 2000 (with the proceedings published in *Lecture Notes in Computer Science*, volume 2235), in August 2001 (with a selection of papers published as a special issue of *Fundamenta Informaticae*, volume 49, numbers 1–3, 2002), and in August 2002 (with the proceedings published in *Lecture Notes in Computer Science*, volume 2597). The fourth and the fifth workshops were organized in Tarragona, Spain, in July 2003, and in Milan, Italy, in June 2004, with the proceedings published as volumes 2933 and 3365, respectively, of *Lecture Notes in Computer Science*.

The pre-proceedings of WMC6 were published by the Institute for Computer Languages of the Vienna University of Technology, and they were available during the workshop. Conforming with tradition, this workshop, too, was a lively scientific event, with many questions and engaged discussions following presentations of papers.

The current volume is based on a selection of papers from the pre-proceedings. These papers were significantly modified according to the discussions that took place during the workshop, and all the selected papers were additionally refereed. The papers in this volume cover all the main directions of research in membrane computing, ranging from theoretical topics in mathematics and computer science, to application issues, especially in biology. More specifically, these papers present research on topics such as: computational power and complexity classes, new types of P systems, relationships to Petri nets, quantum computing, and brane calculi, determinism vs. nondeterminism, hierarchies, the size of small families, algebraic approaches, and designing polynomial solutions to **NP**-complete problems through the use of membrane systems. Like the previous workshops, the scientific program of WMC6 included invited lectures by leading researchers in membrane computing (all the invited talks are represented in this volume) as well as contributed talks based on refereed papers. Altogether, the volume is a faithful illustration of the current state of research in membrane computing (a comprehensive source of information about this fast emerging area of natural computing is the website `http://psystems.disco.unimib.it`).

The workshop was organized by the Institute for Computer Languages of the Vienna University of Technology, under the auspices of the European Molecular Computing Consortium (EMCC).

The Program Committee consisted of Erzsebeth Csuhaj-Varjú (Budapest, Hungary), Rudolf Freund (Vienna, Austria) – Co-chair, Marian Gheorghe (Sheffield, UK), Hendrik Jan Hoogeboom (Leiden, The Netherlands), Oscar H. Ibarra (Santa Barbara, USA), Natasha Jonoska (Tampa, Florida), Kamala Krithivasan

(Madras, India), Vincenzo Manca (Verona, Italy), Maurice Margenstern (Metz, France), Gheorghe Păun (Bucharest, Romania, and Seville, Spain) – Co-chair, Mario J. Pérez-Jiménez (Seville, Spain), Grzegorz Rozenberg (Leiden, The Netherlands, and Boulder, Colorado, USA), Petr Sosík (Opava, Czech Republic), and Claudio Zandron (Milan, Italy).

The editors are indebted to the participants of WMC6 and in particular to the contributors of this volume. Special thanks go to Springer for the efficient cooperation in the timely production of this volume.

November 2005

Rudolf Freund
Gheorghe Păun
Grzegorz Rozenberg
Arto Salomaa

Table of Contents

Invited Lectures

Regular Presentations

Computational Power of Symport/Antiport: History, Advances, and Open Problems

Artiom Alhazov[1,2], Rudolf Freund[3], and Yurii Rogozhin[2]

[1] Research Group on Mathematical Linguistics,
Rovira i Virgili University, Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
artiome.alhazov@estudiants.urv.es
[2] Institute of Mathematics and Computer Science
of the Academy of Sciences of Moldova,
Str. Academiei 5, Chişinău, Moldova
{artiom, rogozhin}@math.md
[3] Faculty of Informatics, Vienna University of Technology,
Favoritenstr. 9–11, A–1040 Vienna, Austria
rudi@emcc.at

Abstract. We first give a historical overview of the most important results obtained in the area of P systems and tissue P systems with *symport/antiport* rules, especially with respect to the development of computational completeness results improving descriptional complexity parameters. We consider the number of membranes (cells in tissue P systems), the weight of the rules, and the number of objects. Then we establish our newest results: P systems with only one membrane, symport rules of weight three, and with only seven additional objects remaining in the skin membrane at the end of a halting computation are computationally complete; P systems with minimal cooperation, i.e., P systems with symport/antiport rules of size one and P systems with symport rules of weight two, are computationally complete with only two membranes with only three and six, respectively, superfluous objects remaining in the output membrane at the end of a halting computation.

1 Introduction

P systems with *symport/antiport* rules, i.e., P systems with *pure communication rules assigned to membranes*, were introduced in [38]. Symport rules move objects across a membrane together in one direction, whereas antiport rules move objects across a membrane in opposite directions. These operations are very powerful, i.e., P systems with symport/antiport rules have universal computational power with only one membrane, e.g., see [15], [22], [17].

After establishing the necessary definitions, we first give a historical overview of the most important results obtained in the area of P systems and tissue P systems with *symport/antiport* rules and review the development of computational completeness results improving descriptional complexity parameters, especially concerning the number of membranes and cells, respectively, and the weight of

R. Freund et al. (Eds.): WMC 2005, LNCS 3850, pp. 1–30, 2006.

the rules as well as the number of objects. Moreover, we establish our newest results: first we prove that P systems with only one membrane and symport rules of weight three can generate any Turing computable set of numbers with only seven additional symbols remaining in the skin membrane at the end of a halting computation, which improves the result of [21] where thirteen superfluous symbols remained. Then we show that P systems with minimal cooperation, i.e., P systems with symport/antiport rules of weight one and P systems with symport rules of weight two, are computationally complete with only two membranes modulo some initial segment. In P systems with symport/antiport rules of weight one, only three superfluous objects remain in the output membrane at the end of a halting computation, whereas in P systems with symport rules of weight two six additional objects remain. For both variants, in [5] it has been shown that two membranes are enough to obtain computational completeness modulo a terminal alphabet; in this paper, we now show that the use of a terminal alphabet can be avoided for the price of superfluous objects remaining in the output membrane at the end of a halting computation. So far we were not able to completely avoid these additional objects, hence, it remains as an interesting question how to reduce their number.

2 Basic Notions and Definitions

For the basic elements of formal language theory needed in the following, we refer to [45]. We just list a few notions and notations: $\mathbb{N}$ denotes the set of natural numbers (i.e., of non-negative integers). V^* is the free monoid generated by the alphabet V under the operation of concatenation and the empty string, denoted by λ, as unit element; by $\mathbb{N}RE$, $\mathbb{N}REG$, and $\mathbb{N}FIN$ we denote the family of recursively enumerable sets, regular sets, and finite sets of natural numbers, respectively. For $k \geq 1$, by $\mathbb{N}_k RE$ we denote the family of recursively enumerable sets of natural numbers excluding the initial segment 0 to $k-1$. Equivalently, $\mathbb{N}_k RE = \{k + L \mid L \in \mathbb{N}RE\}$, where $k + L = \{k + n \mid n \in L\}$.

Let $\{a_1, \ldots, a_n\}$ be an arbitrary alphabet; the number of occurrences of a symbol a_i in x is denoted by $|x|_{a_i}$; the *Parikh vector* associated with x with respect to $a_1, \ldots, a_n$ is $\left(|x|_{a_1}, \ldots, |x|_{a_n}\right)$. The *Parikh image* of a language L over $\{a_1, \ldots, a_n\}$ is the set of all Parikh vectors of strings in L. A (finite) multiset $\langle m_1, a_1\rangle \ldots \langle m_n, a_n\rangle$ with $m_i \in \mathbb{N}$, $1 \leq i \leq n$, can be represented by any string x the Parikh vector of which with respect to $a_1, \ldots, a_n$ is $(m_1, \ldots, m_n)$.

The family of recursively enumerable sets of vectors of natural numbers is denoted by $PsRE$.

2.1 Register Machines and Counter Automata

The proofs of the main results discussed in this paper are based on the simulation of register machines or counter automata, respectively; with respect to register machines, we refer to [37] for original definitions, and to [13] for definitions like those we use in this paper.

A (non-deterministic) *register machine* is a construct

$$M = (d, Q, q_0, q_f, P),$$

where:

- d is the number of registers,
- Q is a finite set of label for the instructions of M,
- q_0 is the initial label,
- q_f is the final label, and
- P is a finite set of instructions injectively labelled with elements from Q. The labelled instructions are of the following forms:
  1. $q_1 : (A(r), q_2, q_3)$;
     add 1 to the contents of register r and proceed to one of the instructions (labelled with) q_2 and q_3 ("ADD"-instruction).
  2. $q_1 : (S(r), q_2, q_3)$;
     if register r is not empty, then subtract 1 from its contents and go to instruction q_2, otherwise proceed to instruction q_3 ("SUBTRACT"-instruction).
  3. $q_f : halt$;
     stop the machine; the final label q_f is only assigned to this instruction.

A (non-deterministic) register machine M is said to generate a vector of natural numbers $(s_1, \ldots, s_k)$ if, starting with the instruction with label q_0 and all registers containing the number 0, the machine stops (it reaches the instruction $q_f : halt$) with the first k registers containing the numbers $s_1, \ldots, s_k$ (and all other registers being empty).

The register machines are known to be computationally complete, equal in power to (non-deterministic) Turing machines: they generate exactly the sets of vectors of natural numbers which can be generated by Turing machines, i.e., the family $PsRE$. More precisely, from the main result in [37] that the actions of a Turing machine can be simulated by a register machine with two registers (using a prime number encoding of the configuration of the Turing machine) we know that any recursively enumerable set of k-vectors of natural numbers can be generated by a register machine with $k+2$ registers where only "ADD"-instructions are needed for the first k registers.

A non-deterministic *counter automaton* is a construct

$$M = (d, Q, q_0, q_f, P),$$

where:

- d is the number of counters, and we denote $D = \{1, \ldots, d\}$;
- Q is a finite set of states, and without loss of generality, we use the notation $Q = \{q_i \mid 0 \leq i \leq f\}$ and $F = \{0, 1, \ldots, f\}$,
- $q_0 \in Q$ is the initial state,
- $q_f \in Q$ is the final state, and
- P is a finite set of instructions of the following forms:

1. $(q_i \to q_l, k+)$, with $i, l \in F,\ i \neq f,\ k \in D$ ("increment"-instruction). This instruction increments counter k by one and changes the state of the system from q_i to q_l.
2. $(q_i \to q_l, k-)$, with $i, l \in F,\ i \neq f,\ k \in D$ ("decrement"-instruction). If the value of counter k is greater than zero, then this instruction decrements it by 1 and changes the state of the system from q_i to q_l. Otherwise (when the value of register k is zero) the computation is blocked in state q_i.
3. $(q_i \to q_l, k = 0)$, with $i, l \in F,\ i \neq f,\ k \in D$ ("test for zero"-instruction). If the value of counter k is zero, then this instruction changes the state of the system from q_i to q_l. Otherwise (the value stored in counter k is greater than zero) the computation is blocked in state q_i.
4. *halt*. This instruction stops the computation of the counter automaton, and it can only be assigned to the final state q_f.

A transition of the counter automaton consists in updating/checking the value of a counter according to an instruction of one of the types described above and by changing the current state to another one. The computation starts in state q_0 with all counters being equal to zero. The result of the computation of a counter automaton is the value of the first k counters when the automaton halts in state $q_f \in Q$ (without loss of generality we may assume that in this case all other counters are empty). A counter automaton thus (by means of all computations) generates a set of k-vectors of natural numbers. As for register machines, we know that any set of k-vectors of natural numbers from $PsRE$ can be generated by a counter automaton with $k+2$ counters where only "increment"-instructions are needed for the first k counters.

A special variant of counter automata uses a set C of pairs $\{i, j\}$ with $i, j \in Q$ and $i \neq j$. As a part of the semantics of the *counter automaton with conflicting counters* $M = (d, Q, q_0, q_f, P, C)$, the automaton stops without yielding a result whenever it reaches a configuration where, for any pair of conflicting counters, both are non-empty.

Given an arbitrary counter automaton, we can easily construct an equivalent counter automaton with conflicting counters: For every counter i which shall also be tested for zero, we add a conflicting counter $\bar{\imath}$; then we replace all "test for zero"-instructions $(l \to l', i = 0)$ by the sequence of instructions $(l \to l'', \bar{\imath}+)$, $(l'' \to l', \bar{\imath}-)$. Thus, in counter automata with conflicting counters we only use "increment"-instructions and "decrement"-instructions, whereas the "test for zero"-instructions are replaced by the special conflicting counters semantics.

Another special variant of a counter automaton is called *partially blind (multi) counter automaton* (or machine, [23]); we shall use the abbreviation PBCA for this restricted type of counter automata which consists of a finite number (we call the number m) of counters that can add one and subtract one, but cannot test for zero. If there is an attempt to decrement a zero counter, the system aborts and does not accept. The first k counters (for some $k \leq m$) are input counters. The system is started with some nonnegative integers $(n_1, \ldots, n_k)$ in the input counters and the other counters set to zero. The input tuple is accepted if the

system reaches a halting state and all the counters are zero. Hence, the language accepted by a PBCA is the set of k-tuples of nonnegative integers accepted by the system.

Formally a PBCA is defined as $M = (m, B, l_0, l_h, R)$ where m is the number of partially blind counters in the system, B is the set of instruction labels, l_0 is the starting instruction, l_h is the halting instruction, and R is the set of labelled instructions. These labelled instructions in R are of the forms:

- $l_i : (ADD(r), l_j)$,
- $l_i : (SUB(r), l_j)$,
- $l_i : HALT$,

where l_i and l_j are instruction labels and r is the counter that should be added/ subtracted.

For notational convenience, we will denote the family of sets of tuples of natural numbers accepted by some PBCA as $aPBLIND$ and the family of sets of tuples of natural numbers accepted by PBCAs with m counters as m-$aPBLIND$.

A related model called *blind (multi)counter automaton* (or machine, see [23]) is a (multi)counter automaton that can add one and subtract one from a counter, but cannot test a counter for zero. The difference between this model and a partially blind counter automaton is that a blind counter automaton does not abort when a zero counter is decremented. Thus, the counter can store negative numbers. Again, an input is accepted if the computation reaches an accept state and all the counters are zero.

We note that blind counter automata are equivalent in power to reversal bounded counter automata [23] which are equivalent to semilinear sets [30]. Partially blind counter automata are strictly more powerful than blind counter automata [23].

We have defined a PBCA as an acceptor for k-tuples of nonnegative integers. One can also define a partially blind counter automaton that is used as a generator of k-tuples of nonnegative integers [29]. A *partially blind counter generator* (PBCG) M consists of m counters, where the first $k \leq m$ counters are distinguished as the *output counters*. M starts with all counters set to zero. Again, at each step, each counter can be incremented/decremented by 1 (or left unchanged), but if there is an attempt to decrement a zero counter, the system aborts and does not generate anything. If the system halts in a final state with zero in counters $k+1, \ldots, m$, then the tuple $(n_1, \ldots, n_k)$ in the first k counters is said to be generated by M.

A restricted variant of a counter automaton is called *linear-bounded multicounter automaton* (or machine).

A *deterministic* multicounter automaton Z is linear-bounded if, when given an input n in one of its counters (called the input counter) and zeros in the other counters, it computes in such a way that the sum of the values of the counters at any time during the computation is at most n. One can easily normalize the computation so that every increment is preceded by a decrement (i.e., if Z