Journal Subline

LNBI 3939

Transactions on Computational Systems Biology IV

Corrado Priami
Editor-in-Chief

Springer

Corrado Priami Luca Cardelli
Stephen Emmott (Eds.)

Transactions on Computational Systems Biology IV

Springer

Series Editors

Sorin Istrail, Brown University, Providence, RI, USA
Pavel Pevzner, University of California, San Diego, CA, USA
Michael Waterman, University of Southern California, Los Angeles, CA, USA

Volume Editors

Corrado Priami
The Microsoft Research - University of Trento
Centre for Computational and Systems Biology
Piazza Manci, 17, 38050 Povo (TN), Italy
E-mail: priami@msr-unitn.unitn.it

Luca Cardelli
Stephen Emmott
Microsoft Research Cambridge
7 JJ Thomson Avenue, Cambridge CB3 0FB, UK
E-mail: {luca,semmott}@microsoft.com

Library of Congress Control Number: 2006923001

CR Subject Classification (1998): J.3, H.2.8, F.1

LNCS Sublibrary: SL 8 – Bioinformatics

ISSN 0302-9743
ISBN-10 3-540-33245-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-33245-9 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2006
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11732488 06/3142 5 4 3 2 1 0

Lecture Notes in Bioinformatics 3939

Edited by S. Istrail, P. Pevzner, and M. Waterman

Editorial Board: A. Apostolico S. Brunak M. Gelfand
T. Lengauer S. Miyano G. Myers M.-F. Sagot D. Sankoff
R. Shamir T. Speed M. Vingron W. Wong

Subseries of Lecture Notes in Computer Science

Lecture Notes in Bioinformatics

Vol. 3939: C. Priami, L. Cardelli, S. Emmott (Eds.), Transactions on Computational Systems Biology IV. VII, 141 pages. 2006.

Vol. 3916: J. Li, Q. Yang, A.-H. Tan (Eds.), Data Mining for Biomedical Applications. VIII, 155 pages. 2006.

Vol. 3909: A. Apostolico, C. Guerra, S. Istrail, P.A. Pevzner, M. Waterman (Eds.), Research in Computational Molecular Biology. XVII, 612 pages. 2006.

Vol. 3886: E.G. Bremer, J. Hakenberg, E.-H.(S.) Han, D. Berrar, W. Dubitzky (Eds.), Knowledge Discovery in Life Science Literature. XIV, 147 pages. 2006.

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

Vol. 3737: C. Priami, E. Merelli, P. Gonzalez, A. Omicini (Eds.), Transactions on Computational Systems Biology III. VII, 169 pages. 2005.

Vol. 3695: M.R. Berthold, R.C. Glen, K. Diederichs, O. Kohlbacher, I. Fischer (Eds.), Computational Life Sciences. XI, 277 pages. 2005.

Vol. 3692: R. Casadio, G. Myers (Eds.), Algorithms in Bioinformatics. X, 436 pages. 2005.

Vol. 3680: C. Priami, A. Zelikovsky (Eds.), Transactions on Computational Systems Biology II. IX, 153 pages. 2005.

Vol. 3678: A. McLysaght, D.H. Huson (Eds.), Comparative Genomics. VIII, 167 pages. 2005.

Vol. 3615: B. Ludäscher, L. Raschid (Eds.), Data Integration in the Life Sciences. XII, 344 pages. 2005.

Vol. 3594: J.C. Setubal, S. Verjovski-Almeida (Eds.), Advances in Bioinformatics and Computational Biology. XIV, 258 pages. 2005.

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

Vol. 3388: J. Lagergren (Ed.), Comparative Genomics. VII, 133 pages. 2005.

Vol. 3380: C. Priami (Ed.), Transactions on Computational Systems Biology I. IX, 111 pages. 2005.

Vol. 3370: A. Konagaya, K. Satou (Eds.), Grid Computing in Life Science. X, 188 pages. 2005.

Vol. 3318: E. Eskin, C. Workman (Eds.), Regulatory Genomics. VII, 115 pages. 2005.

Vol. 3240: I. Jonassen, J. Kim (Eds.), Algorithms in Bioinformatics. IX, 476 pages. 2004.

Vol. 3082: V. Danos, V. Schachter (Eds.), Computational Methods in Systems Biology. IX, 280 pages. 2005.

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

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

Vol. 2812: G. Benson, R.D. M. Page (Eds.), Algorithms in Bioinformatics. X, 528 pages. 2003.

Vol. 2666: C. Guerra, S. Istrail (Eds.), Mathematical Methods for Protein Structure Analysis and Design. XI, 157 pages. 2003.

此为试读,需要完整PDF请访问: www.ertongbook.com

Preface

This issue of the journal reports some selected contributions from the First Converging Science conference held in Trento in December 2004 and chaired by Luca Cardelli, Stephen Emmott and Corrado Priami. The first contribution is the transcription of the keynote lecture by Robin Milner on the foundations of global computing. The second paper, by John Wooley, reports on the interdisciplinarity in innovation initiatives. Ronan Sleep presents a grand challenge for the convergence of sciences in the third paper, while Andrew Jones discusses how to apply computer science research to biodiversity in the fourth contribution. The fifth paper, by Bob Hertzberger, introduces the concept of e-science. The sixth paper, by Francois Fages, describes the role of syntax and semantics of programming languages in systems biology, while Ivan Arisi and his co-authors report on a European initiative on neuronal cells in the seventh contribution. The eighth paper, by Imrich Chlamtac et al., deals with the role of biological inspiration in autonomic computing. The ninth paper, by Riguidel, covers digitalization and telecommunications.

The volume ends with two regular contributions. The first one by Blossey, Cardelli and Phillips describes a compositional approach to the quantitative modelling of gene networks. The last paper of the volume by Jeong and Miyano introduces a prediction method for the protein–RNA interaction.

January 2006 Corrado Priami

LNCS Transactions on Computational Systems Biology – Editorial Board

Corrado Priami, Editor-in-chief	University of Trento, Italy
Charles Auffray	Genexpress, CNRS and Pierre & Marie Curie University, France
Matthew Bellgard	Murdoch University, Australia
Soren Brunak	Technical University of Denmark, Denmark
Luca Cardelli	Microsoft Research Cambridge, UK
Zhu Chen	Shanghai Institute of Hematology, China
Vincent Danos	CNRS, University of Paris VII, France
Eytan Domany	Center for Systems Biology, Weizmann Institute, Israel
Walter Fontana	Santa Fe Institute, USA
Takashi Gojobori	National Institute of Genetics, Japan
Martijn A. Huynen	Center for Molecular and Biomolecular Informatics, The Netherlands
Marta Kwiatkowska	University of Birmingham, UK
Doron Lancet	Crown Human Genome Center, Israel
Pedro Mendes	Virginia Bioinformatics Institute, USA
Bud Mishra	Courant Institute and Cold Spring Harbor Lab, USA
Satoru Miayano	University of Tokyo, Japan
Denis Noble	University of Oxford, UK
Yi Pan	Georgia State University, USA
Alberto Policriti	University of Udine, Italy
Magali Roux-Rouquie	CNRS, Pasteur Institute, France
Vincent Schachter	Genoscope, France
Adelinde Uhrmacher	University of Rostock, Germany
Alfonso Valencia	Centro Nacional de Biotecnologa, Spain

Table of Contents

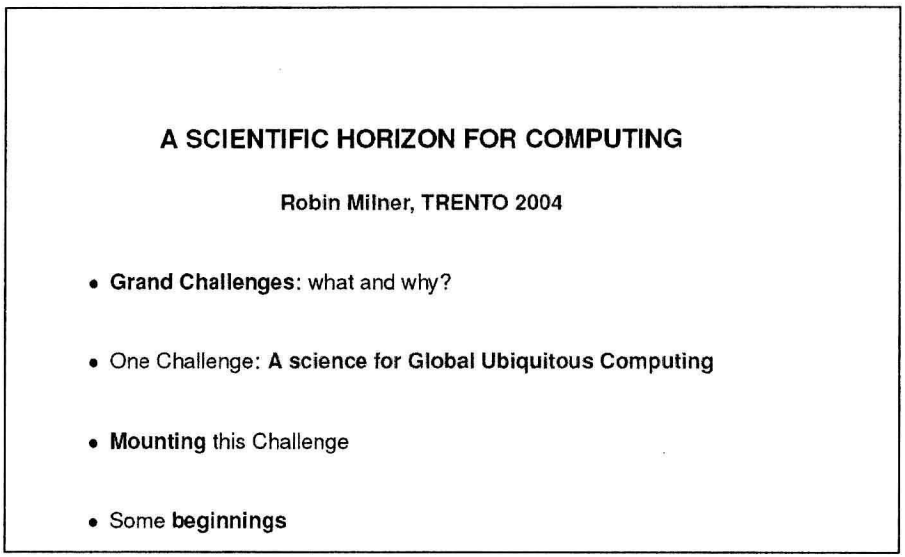
Transcription of the Presentation
Scientific Foundation for Global Computing

Robin Milner

University of Cambridge, UK

It is a big honour to be able to speak at one of the most exciting conferences I have been to for 20 years.

I feel in some ways daunted because, although the development of connections between biology and computer science may seem wonderful from outside, we know that they depend crucially on details. We have to believe with confidence that what we have already done justifies bringing the subjects together in this way. I think that's we are going to see in the later talks. I want to try to anticipate a little bit of that here. But first I want to talk about global computing. We may describe it, fancifully perhaps, as the arrival of a single global computer, which is increasingly pervading lives.



The sequence in my talk will be this: first I am going to put the global computer in the context of the UK exercise on Grand Challenges for Research in Computing, to create a framework for it. Secondly, I am going to focus on a specific Challenge: to build a science that will underlie all this pervasive computing. (We would like to trust it; but are we ready to trust computing systems on such a large scale?) Thirdly: How shall we put more structure into developing this science? What ingredients do we

C. Priami et al. (Eds.): Trans. on Comput. Syst. Biol. IV, LNBI 3939, pp. 1 – 13, 2006.
© Springer-Verlag Berlin Heidelberg 2006

already have? And fourthly, I am going to talk about some beginnings for this science; this gives me a sense of security, as it indicates that we are starting from somewhere.

WHAT IS A GRAND CHALLENGE EXERCISE?

- The community examines and adopts **long-term goals** ...
- ...from **within the science**, not outside it.
- Thus to develop and refine a **portfolio of proposals** ...
- ...showing the public (and funders) **what we aspire to**.

What is a Grand Challenges Exercise? We embark upon it because we believe that long-term challenges for computer science can't just be written down on a piece of paper in five minutes; you have to work towards them. You have to bring together some ingredients into a goal which is so well-focussed that you can devise a plan to reach that goal over 15 years. This focussing takes time, perhaps five years, and it has to come from the community - it can't be dictated from above. The community examines a portfolio of *proposals* for long-term challenges; some of them are successfully developed into Grand Challenges – something of which we can say "Yes! We can *plan* to achieve this in 15 years." Of course it still has a reasonable possibility of failure, indeed, the plan must define what would count as failure.

UK PROPOSALS for
GRAND CHALLENGES IN COMPUTING

1 **IVIS: In Vivo ⇔ In Silico**

2 **Science for Global Ubiquitous Computing**

3 **Memories for Life**

4 **Scalable Ubiquitous Computing Systems**

5 **Architecture for Brain and Mind**

6 **Dependable Systems Evolution**

7 **Journeys in Non-Classical Computation**

Such an exercise helps us to identify, within our subject community, what our real aims are; it also has the benefit of showing the public and the funding agencies what we aspire to. We must cease to think just in terms of building the next technology. We must have aspirations, defining the directions we wish to take.

So we have developed a portfolio of Grand Challenge proposals. Rather conveniently, two among them are perfectly relevant to what we shall be discussing in these two days. The first, *In Vivo--In Silico*, has to do with bringing computing into biology or vice versa. It is co-ordinated by Ronan Sleep, who is here with us; he has asked me to say that there are some handouts to do with this Grand Challenge at the desk outside. So that is the first Challenge on our list; it wouldn't exist if it were not for the success of some small but very indicative predictive experiments, bringing computing models into biology.

The second one, *Science for Global Ubiquitous Computing*, is the one I want to start from. I want to start there because I know something about how to predict what should happen there, but also because one of the things we would like to see is a convergence between the models used on the one hand to *engineer* pervasive computing, and those used on the other hand to *understand* existing scientific -- possibly biological -- systems. It's almost too much to hope that the principles for those two things are the same. But let us at least entertain that hypothesis, because this is one of the most exciting things that could happen. We see such disparate fields of study as possibly having the same underlying very particular primitive ideas. So I want to have time to say something about that later in my talk.

SCIENCE FOR GLOBAL UBIQUITOUS COMPUTING

- By 2020, a single **Global Ubiquitous Computer (GUC)**
- Part designed, part natural phenomenon
- Shall we understand it?

First then, what about the *Science for Global Ubiquitous Computing*? You could imagine we have a single computer, or network of computing entities, that pervades the world by 2020. Who's to say that it will not happen? It is going to be partly designed and partly a natural phenomenon, because it comes together from the energies of different people collaborating or possibly just simply doing their own thing and then joining their systems up. The question is: Shall we understand it? We have problems understanding our own software. For example, recently in UK we have the glorious failure of the government's Child Support Agency to produce a decent computing system. Grand failures like this are common.

此为试读,需要完整PDF请访问: www.ertongbook.com

UNDERSTANDING and BUILDING

- Underlying both are **modelling kits**
- **Traditional science and engineering** has
  Differential equations, Laplace transforms, Matrix algebra, ...
  ... and they join understanding with building
- **Computer science and engineering** has
  Automata, Languages, Relational algebra, Network theories, Logics, Stochastics, Type theory, Process calculi, Semi-structured data, ...
  ... but the junction is tenuous. **Why?**
- For **Ubiquity??** Separation will lead to *stagnation or worse.*

So the key, for me, is that understanding and building have to go together. Both science and engineering involve modelling kits. In standard mathematics and engineering it is easy to reel off a list of these things: differential equations, Laplace transforms etc. Computer science also has a number of theoretical models, which could claim this kind of status; but the fact is that those theories are not really used in the engineering of computing systems. Why not? Well, the theories struggle along trying to keep up with the latest technology. I think that's why it is difficult for the theories to inform engineering practice. Our technologies, both hardware and software, move so fast and demand new ideas in such a way that new theories have to emerge to try to keep up with them. So, even now, the junction is tenuous between the science and the engineering of computing systems; this situation will be amplified as we move forward into global computing. But the continued separation between science and engineering will lead to stagnation of systems, or even to disaster.

The Challenge: SCIENCE FOR GLOBAL UBIQUITOUS COMPUTING

- *To develop an informatic science whose* **concepts, calculi, theories** *and automated* **tools** *allow descriptive and predictive analysis of the GUC at each level of* **abstraction**
- *That every* **system** *and* **software** *construction —including languages— for the GUC shall employ only these concepts and calculi, and be* **analysed** *and* **justified** *by these theories and tools.*

`www.nesc.ac.uk/esi/events/Grand_Challenges/proposals/Ubiq.pdf`

An *ideal* goal? But no argument limits the degree of possible success!

In our proposal we have boldly formulated a couple of statements which should be the Grand Challenge for global computing:

- *To develop an informatic science whose concepts, calculi, theories and automatic tools allow descriptive and predictive analysis of the global ubiquitous computer at each level of abstraction.*
- *That every system and every software construction, including languages for global computing, shall employ only those concepts and calculi, and shall be analysed and justified by those theories and tools.*

That is the goal that we want realise. It is an ideal goal, but there is no argument that limits the degree of possible success in that direction. That is the important point: not that we won't fail, but there is no present reason that you could claim *why* we should fail, except for dragging our feet.

A THEORETICAL HIERARCHY

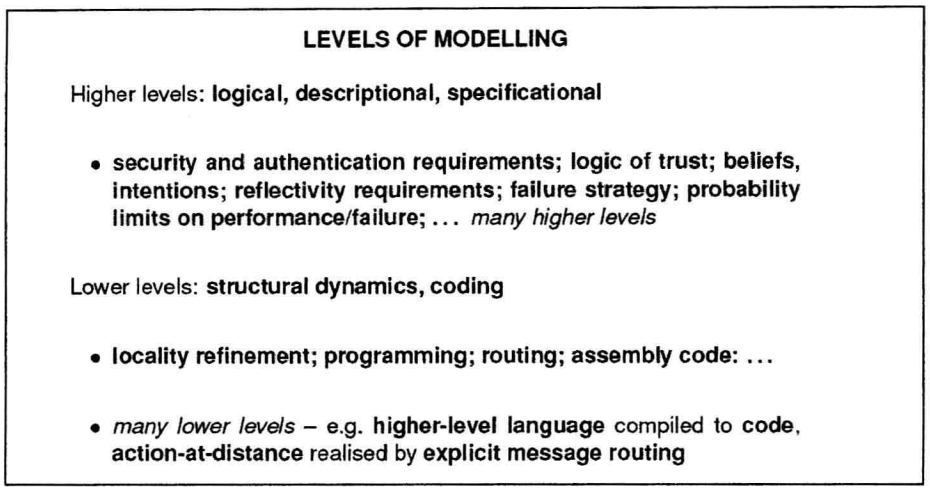
Theoretical goals for the Grand Challenge:

- *To express theories for the GUC as a hierarchy of* **models** *and* **languages**, *assigning each concept (*e.g. *trust) to a certain level in the hierarchy*
- *To define, for each model M, how a system description in M may be* **realised** *or* **implemented** *in models/languages $M_1, \ldots, M_n$ lying below M*

Why do we need models at many abstraction levels?

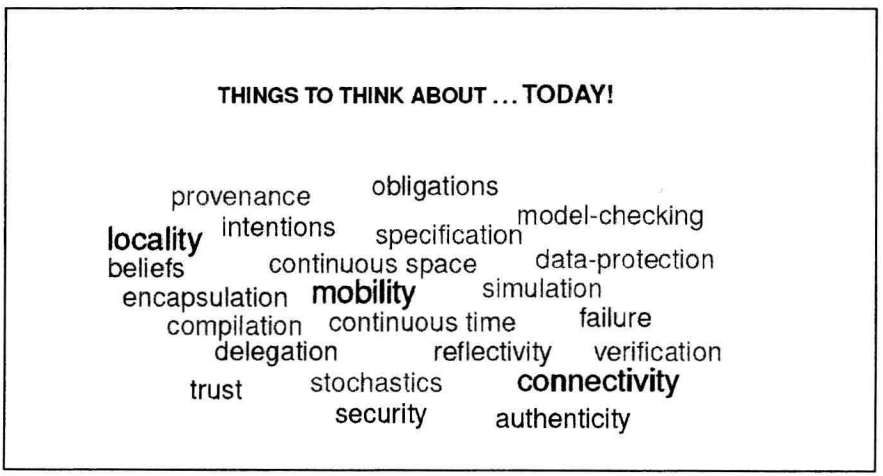
The theory behind this will have to build a hierarchy of models and languages; it will somehow assign each concept, such as *trust* or *failure* or *interactivity* or *logical analysis* or *software development*, to a certain level in the hierarchy. Then, at each level of modelling, it must define how a system described at that level can be realised or implemented in terms of languages or models which lie below it. That is of course a very clean picture; it is never going to look quite like that. But a hierarchy of models is going to be essential, because we shall be dealing with a huge population -- perhaps billions – of computing entities.

We need many abstraction levels. I am not going to go into great detail about them, but they must follow a pattern. First of all, at a higher level we tend to be descriptional; we tend to specify how systems should behave; we tend to analyse them logically, often in terms of rather high level entities like trust or in terms of certain invariant properties that represent acceptable behaviour. At intermediate levels there might be strategies for failure management, probability limits on performance of failure, reflectivity requirements (the ability for a system to report on what it has



done). At the lower levels we have structural dynamics, how things actually work in practice -- including machine code, routing of messages and so on. So you can see the kind of things that this levelling consists of. I don't believe that we can build a theory without such levels. I even believe that there should be a fractal quality here; for example, movement of data within a computer program should be treated in the same way as movement of agents in a natural or built environment. There should be a certain repetition of concepts among the levels; how else can we understand a huge organism with a manageable repertoire of concepts?

I think that's enough about levels of modelling.



This slide shows just some of the concepts we have to consider, things like data protection, or the intentions of a individual human or software agent, or delegation of work from one system to another. We also have to bring in an understanding of continuous time and space, and our analysis must have a probabilistic or stochastic

element. You see this huge space of things. I want to pick out three things that seem to lie where our collaboration with biology begins, and also where we begin to understand global computing. They are: *locality, connectivity,* and *mobility.*

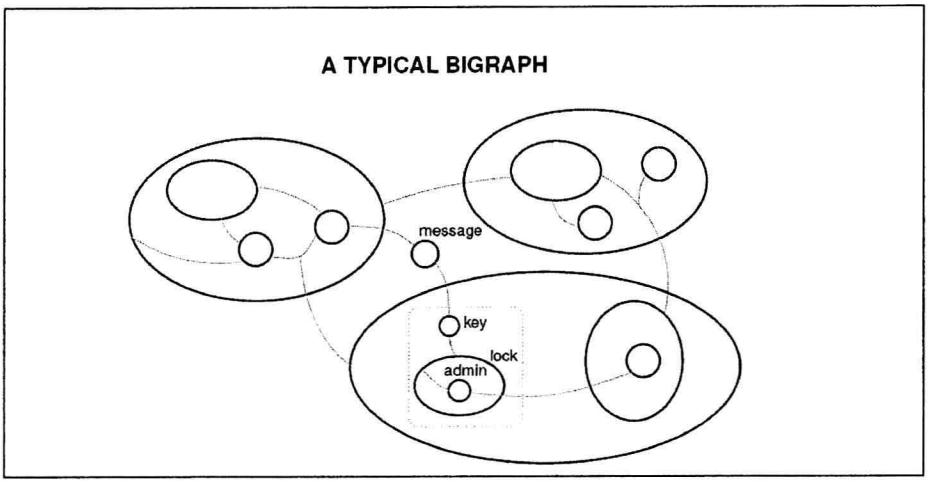
A BEGINNING: STRUCTURAL DYNAMICS

- GUC systems reconfigure both their *topography* and their *connectivity*, both physical and virtual.
- Mobile processes can be modelled by **pi-calculus** and by **mobile ambients** ...
- ... so try using **bigraphs**, which generalise these.
- Then extend to a *stochastic* model with *continuous time and space* ...
- ... both for *modelling* and for *programming*.

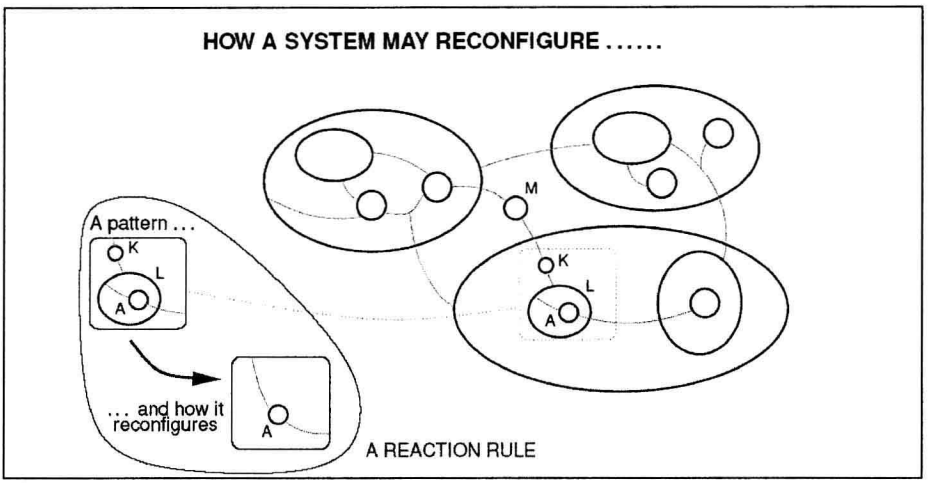
Locality, or topography, is of course an essential feature of computing systems that may be embedded in our environment or even in our bodies. It may appear to be a new concern in computing, but in fact we have always used topographical intuitions in our computer programs. For example, we talk of storage *space,* or *local* variables. It even looks as though the macroscopic or real-life topographic considerations that come with global computing may line up with the microscopic or virtual topography that lives inside a computer program. That would be a wonderful junction. But there is more to this space than just locations. By programming, we achieve a situation in which entities located far apart may nevertheless be connected; their means of connection may be ultimately a matter of physical fibres or wireless, but at an abstract level we think of these distant neighbours as if they were next-door neighbours; that is how we understand accessing remotely located websites, for example. Thus *connectivity* can be usefully seen as independent of locality.

To complete the trio, a system may freely reconfigure both its localities and its connectivity; that is what we mean by *mobility.* Do we get the same kinds of mobility in physical and in virtual space? Do we get the same kinds both in artificial (computing) and in natural (biological) systems? If so, we would have a new and exciting theory of dynamical systems, with emphasis upon discrete entities. Moreover, we would hope to apply these ideas equally at both high and low (abstract and concrete) levels.

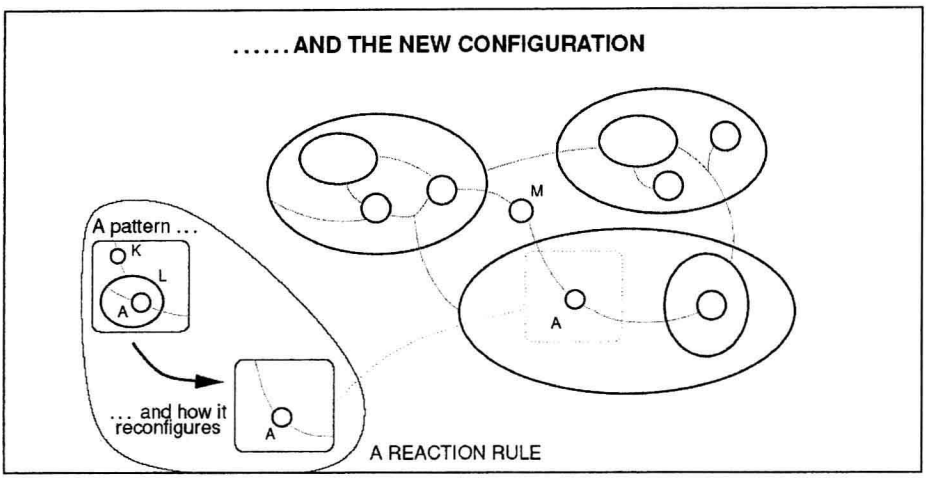
Recently, two calculi of discrete dynamics from computer science have been applied with some success to both biology and pervasive computing; these are the Pi calculus and the calculus of Mobile Ambients. Roughly speaking, they deal respectively with connectivity and with locality. So I am now working with a model called Bigraphs, which tries to capture the best parts of those. The prefix "bi-" refers to the two structural elements: *placing* (locality) and *linking* (connectivity).



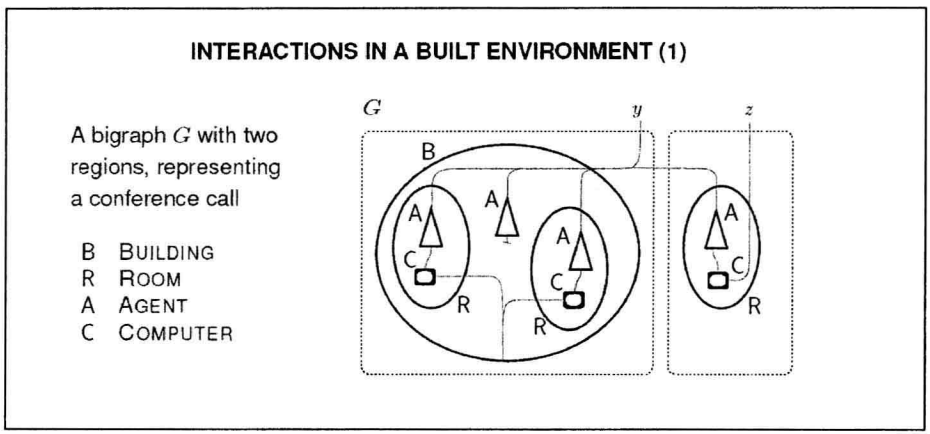
Here is what I call a bigraph. This is a system in which you have things nested inside each other, but they may also communicate across those boundaries irrespective of where they reside. So these green links in the picture connect anything to anything else. I have called some of the nodes *message*, *key* and *lock*, because I wanted to indicate that this might be a system in which a message is trying to insert a key into a lock, in order to find its way to a receiver in the bottom right-hand corner.



In more detail, here is the message **M**, right in the middle of the picture, the key **K** and the lock **L**; and in the bottom left-hand box is what I call a *reaction rule*. Think of it as a very elementary piece of dynamics; it says that whenever you have a pattern of **K** next to **L**, with an agent **A** inside there, then the pattern can change, the lock and key vanish and you are left with the agent **A** ready to help the message **M** further on its journey,
and this slide shows the final state.



Can we use this very elementary kind of dynamics to explain other kinds of interactivity, possibly using different reaction rules? The idea is that we stick to the same kind of structure – bigraphs – but we use different reaction rules to describe what goes on in different kinds of system.



Look at an elementary, but nonetheless somewhat realistic, example of equipping a building with computers that are also sensors. I tried to model what happens when agents, who could be people carrying devices, move around the building. Here is a bigraph *G,* representing the system. **B** is a building, **A** is an agent, **C** a computer and **R** a room, so you have in the picture agents in rooms (and one not in a room but in the building, say in the corridor) connected to computers. The computers themselves are connected to the infrastructure of the building. This picture represents a subsystem of a larger system; it may be situated in a larger host system representing a larger piece of the world.

此为试读,需要完整PDF请访问: www.ertongbook.com