Basic Research in Computer Science

BRICS NS-96-2 Cheng et al. (eds.): BRICS Autumn School on Verification

Programme and Abstracts of the BRICS Autumn School on Verification

October 28 – November 1, 1996, Aarhus, Denmark

Allan Cheng
Kim G. Larsen
Mogens Nielsen (editors)

BRICS Notes Series NS-96-2
ISSN 0909-3206 August 1996

See back inner page for a list of recent publications in the BRICS Notes Series. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK - 8000 Aarhus C
Denmark

Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through World Wide Web and anonymous FTP:

`http://www.brics.dk/`
`ftp://ftp.brics.dk/pub/BRICS`

Programme and Abstracts of the

BRICS Autumn School on Verification

October 28 – November 1, 1996 — Aarhus, Denmark

Allan Cheng
Kim Guldstrand Larsen
Mogens Nielsen
(editors)

How to contact us

For further information, contact:

Allan Cheng/Karen Møller
Department of Computer Science
University of Aarhus
Ny Munkegade, Bldg. 540
DK-8000 Aarhus C
Denmark

e-mail: autumnschool@brics.dk
Tel.: +45 89 42 32 62
Fax: +45 89 42 32 55

BRICS Autumn School on Verification

October 28 – November 1, 1996
Aarhus, Denmark

Dear autumn school participant,

Welcome to BRICS. We hope you will enjoy your stay, the school as well as the surroundings. This note is intended to provide you with helpful information concerning attendance to the school.

Description of BRICS

BRICS, centre for Basic Research In Computer Science, was founded in 1994 by the Danish National Research Foundation at the University of Aarhus and the University of Aalborg; the director is Professor Glynn Winskel. Each year, BRICS focusses on a selected theme in the area of Algorithms and Complexity Theory, Logic, and Semantics.

This year's BRICS theme is Verification. The activities will cover verification of computing systems in a broad sense. More specifically, it has been our intention to investigate specification formalisms, proof principles, and technology of automated tools for verification. Events are taking place during the autumn of '96.

As a special activity, this autumn school will cover the topics

"Theorem Proving and Model Checking"

with special focus on

"Combining deductive methods with model checking techniques"

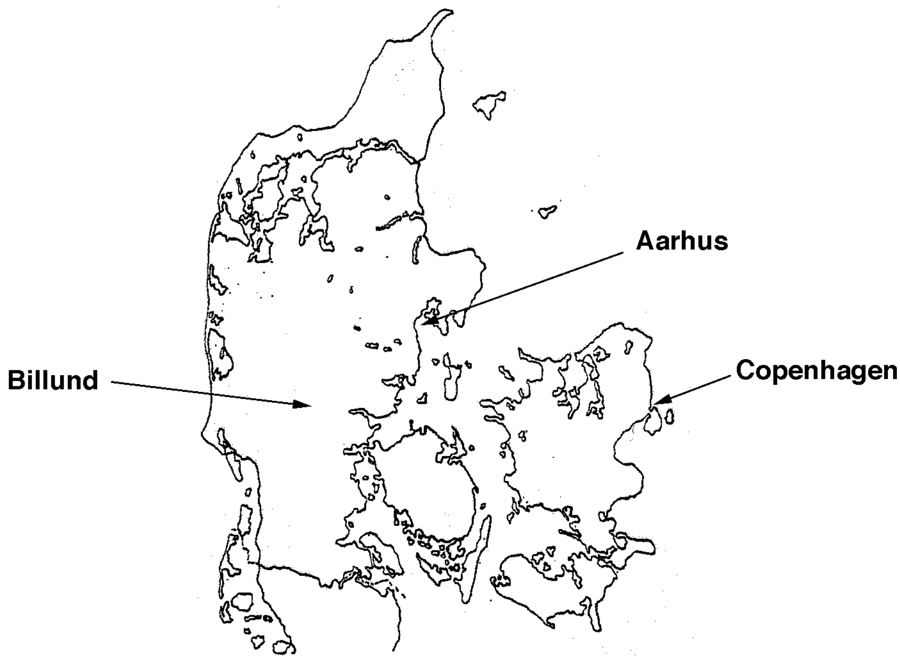
The contents of the autumn school consists of the presentation of a selected set of tools and principles (theorem provers, model checkers and combinations) focussing on "Cutting-Edge Applications-Oriented Techniques". Lectures and presentation of the application of the tools have been weighted somewhat equally.

For more information concerning BRICS, see our WWW-page

`http://www.brics.dk/Activities/96/Verification/index.html`

Travel information

Enclosed with this note is a tourist information booklet. It contains useful information and two maps on page 38 and 39, respectively. In the following we shall refer to them as map 1 and map 2.



Aarhus, Denmark

Arriving by air

Aarhus Airport/Tirstrup (I3, map 1) is connected with Copenhagen International Airport through 11 daily departures on weekdays, five departures on Saturdays and six departures on Sundays. Flying time is 35 minutes. In Copenhagen, a free transit bus links the international and domestic airports (hall 6 for Aarhus). Aarhus Airport/Tirstrup also has direct international flight connections to London, Brussels, Stockholm, Gothenburg and Oslo.

Billund Airport, located 100 kilometres south of Aarhus, has direct international flight connections to Amsterdam, Brussels, Frankfurt, London, Manchester, Birmingham, Oslo, Stavanger, Bergen, Stockholm and Gothenburg.

There is a bus (SAS-bus) service to Aarhus city from both airports. The travel time from Tirstrup to Aarhus is 45 minutes, and the bus costs 50 DKK (Danish Crowns) (one way). This bus will drop you off at the railway station (U18, map 2). The bus (Mærsk-bus) ride from Billund to Aarhus takes approximately 1 1/2 hours, and costs around 120 DKK (one way). The terminal for this bus is the Scandinavian Congress Center (S18, map 2). Tickets are bought on the bus. Individual taxi and limousine services are also available.

Arriving by train

Aarhus has one railway station (T18-U18, map 2).

From Southern Europe there are connections via Hamburg/Flensburg (not Copenhagen). Travel time from Hamburg to Aarhus is about five hours. You may have to change trains.

From Copenhagen comfortable direct InterCity trains depart for Aarhus every hour. Travel time is about four hours and a one way ticket costs approximately 300 DKK, including seat reservation.

Standard inter-regional trains also run every hour with a travel time of about five hours between Copenhagen and Aarhus. If you take such a train, it is necessary to leave the train to embark on the ferry Korsør-Nyborg and enter a new train on the other side of Store Bælt. For these trains no seat reservation is necessary and a one way ticket may cost down to 200 DKK.

Arriving by car

From Southern Europe the route is via Hamburg. Driving time from Hamburg is about four hours by motorway. You should be arriving along E45, see map 1.

Finding your room

Once in Aarhus, you will have to locate the room you have been assigned (see the enclosed cover letter). Below, you will find a list of places and references to map 2.

- PUC (Pressens uddannelsescenter),
  Høegh Guldbergsgade 4,
  phone +45 86136233
  (middle of the right edge of U14, map 2)

- Hotel Ansgar,
  Banegårdspladsen 14,
  phone +45 86124122, fax +45 86202904
  (middle of T18, map 2)

- Eriksens Hotel,
  Banegårdsgade 6–8,
  phone +45 86136296, fax +45 86137676
  (middle/left of T18, map 2)

- Aarhus Sleep-in,
  Havnegade 20,
  phone +45 192055, fax +45 86191811
  (V17, map 2)

Once you have found your way to the counters, just state your name and tell the people at the counter that you are attending the BRICS Autumn School at the University of Aarhus.

Depending on where you stay, there is a 5–20 minute walk to the department of computer science, located at the lower right part of S13, map 2. Fig. 1 gives a more detailed picture of the department.

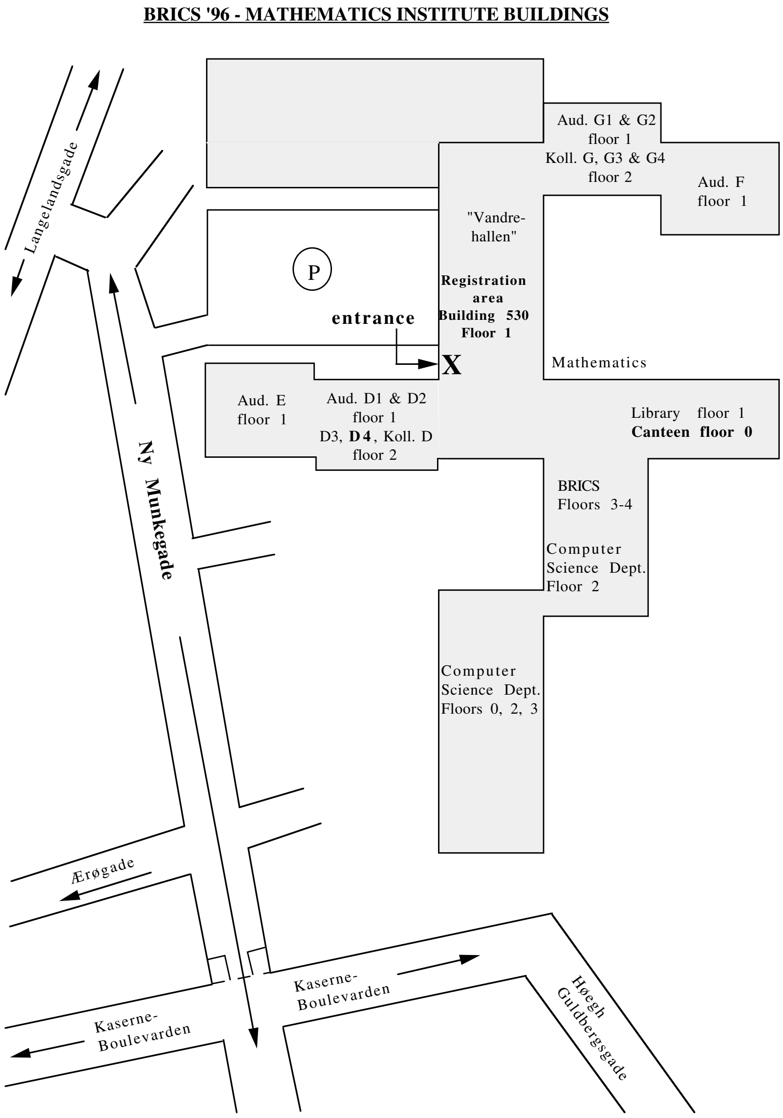


Figure 1

Registration

Registration will take place in "Vandrehallen", see Fig. 1 (this building is located in the lower right part of S13, map 2), Sunday, October 27 from 7:00pm till 9:00pm (snacks, sandwiches, and beverages will be provided), and Monday, October 28 from 8:15pm – 8:50pm.

When registering, you will receive a copy of the lecture notes, a map over the lecture rooms and computing facilities for the school, and a password giving you access to read e-mail and printing.

Scientific Programme

The school has tried to invite distinguished researchers whose research represent important contributions in the field of verification. All lectures will take place at the department, see Fig. 1.

Our invited speakers and their talks are

- **David Basin**,
  Max-Planck-Institut für Informatik
  "Verification Based on Monadic Logic"
- **Ed Clarke**,
  School of Computer Science, Carnegie Mellon University
  "Symbolic Model Checking"
- **Thomas A. Henzinger**,
  University of California at Berkeley
  "Automatic Verification of Real-Time and Hybrid Systems"
- **Gerard Holzmann**,
  Computing Principles Research, Bell Laboratories, USA
  "On-the-Fly Model Checking Tutorial"
- **Tom Melham**,
  Department of Computing Science, University of Glasgow
  "Some Research Issues in Higher Order Logic Theorem Proving"
- **Randy Pollack**,
  Chalmers
  "What we learn from formal checking"

- **Mandayam Srivas**,

  SRI International

  "A Combined Approach to Hardware Verification: Proof-checking, Rewriting with decision procedures and Model-checking"

Abstracts can be found in appendix A. We have tentatively scheduled the lectures as follows[1]:

Time	Monday	Tuesday	Wednesday	Thursday	Friday
09:00–09:50	Holzmann	Holzmann (tool)	Melham	Melham	Srivas
10:00–10:50	Holzmann	Henzinger	Melham	Melham (tool)	Srivas
Coffee break					
11:20–12:10	Holzmann	Henzinger (tool)	Clarke	Pollack	Pollack
12:20–13:10	Basin	Basin	Clarke (tool)	Pollack (tool)	Pollack
Lunch					
14:40–15:30	Basin	Basin (tool)	Excursion	Srivas	Panel disc.
Coffee break					
16:00–16:50	Henzinger	Clarke		Srivas (tool)	Panel disc.
17:00–17:50	Henzinger	Clarke		Larsen (tool)	

[1] (Notice Prof. Kim G. Larsen (BRICS, Aalborg), will be give a tool demo on Thursday).

Every lecturer has four lecturing hours at his disposal. One is tentatively marked with "tool", indicating that a tool demonstration is expected to take place. Tool sessions will take place in the lecture room.

Every day, lunch will be served in the canteen of the Institute of Mathematics, which is situated in the same building as the lecture room on floor 0, see Fig. 1.

Coffee, tea, and cakes will be provided during the afternoon break outside the lecture room D4.

Computing, E-mail, and printing facilities

As a participant of the school you will have access to computing facilities (UNIX-machines). A special username `brics96` is reserved for the participants (the password is handed out during the registration procedure).

For reading e-mail, you may use the usual `rlogin` command or `telnet` command. For printing, you will receive a list of printer names.

Also, local participants will have *badges coloured red*. If ever in doubt about anything, please contact one of these persons. They will either be able to answer your questions or direct you to a person who will.

Social Programme

Wednesday, **October 30**, an excursion to the Museum of prehistory at Moesgaard will take place. Buses will be waiting outside the computer science department ready to leave after lunch.

Upon returning to Århus, late in the afternoon, the buses will drop all participants off at a local restaurant, where tables will be reserved for us.

The excursion and dinner is free (drinks not included), but you are required to register.

Friday, **November 1**, a dinner in the canteen of the Institute of Mathematics will take place to mark the ending of a, hopefully, pleasant week for you.

The dinner is free but you are required to register at the latest Wednesday, October 30.

Some more useful information

The price of DKK can be found on the the WWW-page

`http://www.dr.dk/cgi-bin/ttx/551`

One US dollar costs approximately 5,75 DKK.

The weather in Denmark varies a lot in October. You are advised to bring warm clothes as well as an umbrella or raincoat.

If you should get lost and need to ask your way, don't hesitate to (smile and) ask people; most Danes speak English.

You own notes

Appendix A

TITLE: Verification Based on Monadic Logic
David Basin, Max-Planck-Institut für Informatik

- Automata Theory and Logic
  - Motivation: Decision procedures are useful but limited
  - Contents: Automata Theory & Decision Procedures, Applications, Integration with Higher-Order Logic
- Relationship between Regular Languages, Automata, and Logics
  - survey of history of ideas
  - detailed examination of correspondence for strings
- Verification based on Monadic Logic: the MONA System
- Applications of Decision Procedures
  - I will present examples from verification of parameterized hardware including parameterization in space (e.g., designs operating over data-paths of parameterized length) and time (e.g., sequential circuits).
- Integration with Higher-Order Logic
  - Decision procedures are based on weak decidable logics
  - These logics may be embedded in strong undecidable logics
  - Required are principles for problem decomposition/refinement where decision procedures can be used
    - Refinement based on induction
    - Examples: applications to multipliers and problems on grids

TITLE: Symbolic Model Checking

Ed Clarke, School of Computer Science, Carnegie Mellon University

- Logical errors in finite state reactive systems are an important problem for designers. They can delay getting a new product on the market or cause the failure of some critical device that is already in use. My research group has developed a verification method called temporal logic model checking for this class of systems. In this approach specifications are expressed in a propositional temporal logic, and reactive systems are modeled as state–transition graphs. An efficient search procedure is used to determine automatically if the specifications are satisfied by the state–transition graph. The technique has been used in the past to find subtle errors in a number of non-trivial circuit and protocol designs.

  During the last few years, the size of the reactive systems that can be verified by model checking techniques has increased dramatically. By representing sets of states and transition relations implicitly using Binary Decision Diagrams (BDDs), we are now able to check examples that are many orders of magnitude larger than was previously the case. In this tutorial we describe how the BDD-based model checking techniques work and illustrate their power by discussing the verification of several complex protocol and circuit designs.

TITLE: Automatic Verification of Real-Time and Hybrid Systems

Thomas A. Henzinger, University of California at Berkeley

- Hybrid systems are digital systems that interact with analog environments. Typical examples of hybrid systems are real-time processes (whose analog environment is real-numbered time) and embedded controllers.

  Hybrid systems can be viewed as infinite-state transition systems, and this view gives insights into the structure of hybrid state spaces. For example, for certain interesting classes of hybrid systems, language equivalence, mutual similarity, or bisimilarity induce finite quotients of infinite state spaces. These results can be exploited by automatic analysis techniques for finite-state systems.

  Summary:

  1 Introduction to timed and hybrid automata, a formal model for real-time and hybrid systems; applications from asynchronous circuits, distributed real-time computing, discrete-event control, and robotics

  2 Introduction to HyTech, a symbolic model checking for hybrid automata; parametric, interactive, and approximate verification techniques

  3 Introduction to the theory of timed and hybrid systems: (in)finitary quotient spaces; (un)decidability results

TITLE: On-the-Fly Model Checking Tutorial

Gerard Holzmann, Computing Principles Research, Bell Laboratories, USA

- Abstract: The model checker SPIN is one of the most widely distributed formal verification systems in use today. The system is optimized for on-the-fly verification of linear time temporal logic (LTL) properties of distributed software (not hardware), allowing for dynamically growing or shrinking process systems. The philosophy of verification on which the tool is based differs fundamentally from that of many other systems.

- Overview of topics covered:
  - Scope and Limitations of Tool-Based Verification,
  - Synchronous vs Asynchronous Models,
  - Exhaustive vs Symbolic Model Checking,
  - Process Algebra vs Automata Based Specification,
  - Safety and Liveness Properties, Temporal Logic,
  - Graph Algorithms, Efficiency Considerations,
  - Breadth-First vs Depth-First Search,
  - Nested Depth-First Search,
  - Static vs Dynamic Reduction Methods,
  - Separability of LTL Properties,
  - Real-Time Verification,
  - Proving the Prover, Theorem Proving,
  - Hands-On.

TITLE: Some Research Issues in Higher Order Logic Theorem Proving

Tom Melham, Department of Computing Science, University of Glasgow

- The past twenty years have seen many advances in the use of computers to do logical proofs and in the deployment of this technology in what, broadly speaking, might be called formal methods applications. One important strand has been research in interactive proof-construction in higher-order logics, where the user guides proof discovery and construction but may also employ automatic decision procedures or heuristics. Several well-developed theorem-proving systems (e.g. HOL, Isabelle, Coq, NuPrl) serve as vehicles for this research; and the results of past work have begun to reach industry through, for example, the notable recent achievements in hardware verification using PVS and the use of HOL at several industrial sites.

  This series of talks will focus on three important areas for future research in interactive higher order logic theorem proving:

  - Usability and interface design.
  - Embedding formalisms and embedded theorem proving.
  - Integrating and implementing automated proof methods.

  Concentrating primarily on research issues, I will discuss some past and current work in these areas and outline what I see as some important directions for the future. The discussion will be based on the experiences of the HOL research community but will also be relevant to other higher order logic proof systems.

TITLE: What we learn from formal checking

Randy Pollack, Chalmers

- What is the Meaning of a Machine-Checked Proof ?

  There has been considerable discussion about the value of a large machine-checked proof. For example, the famous 1979 paper by De-Millo, Lipton and Perlis argues that machine verifications cannot be assessed through the "social mechanisms of the mathematical community":

  "They cannot acquire credibility gradually, as a mathematical theorem does; one either believes them blindly as a pure act of faith, or not at all."

  I argue against this statement, suggesting a view, and a technical approach, in which belief in machine verifications is based on evidence, which may be built up over time, and requires participation of the readers. I split the question of belief into two parts; asking whether the claimed proof is actually a derivation in the claimed formal system, and then whether what it proves is the claimed theorem. The first question is addressed by independent checking of formal proofs. I discuss how this approach extends the informal notion of proof as surveyable by human readers, and how surveyability of proofs reappears as the issue of feasibility of proof-checking. The second question has no technical answer; I discuss how it appears in verification of computer systems.

- Using Type Theory: An Introduction

  An introduction to the Curry-Howard isomorphism and the use of type theory for representing mathematics and modelling computer systems. I will concentrate on the Extended Calculus of Constructions (ECC) of Luo, and on the tool LEGO for checking proofs in ECC. The main technical topic is the use of inductive definitions to represent data types (booleans, natural numbers, lists, syntax of a programming language, ...) and relations (typing and operational semantics of a programming language, ...).

- Formalization is Not Just Filling In Details

I outline a small example in LEGO, a formal development of basic lambda calculus. Some "obvious" theorems appear to be very difficult to prove, in that their conventional informal proofs are not easily formalizable. I show a method using generalized induction that makes these "obvious" theorems straightfoward. The moral is that a user of formal checking must be willing to solve new problems, that formalization can teach us new things about "well understood" areas, and that using alternative definitions of basic concepts is one way to learn these new things. If time permits we can pursue the lambda calculus example to outline a formal proof of the Church-Rosser theorem.

TITLE: A Combined Approach to Hardware Verification: Proof-checking, Rewriting with decision procedures and Model-checking

Mandayam Srivas, SRI International

- Abstract: The past decade has seen tremendous progress in the application of formal methods for hardware design and verification. Much of the early work was on applying proof checking and theorem proving tools to the modeling and verification of hardware designs. Though these approaches were quite general, the verification process required a significant human input. More recently, there has been a large body of work devoted to the use of model-checking, language-containment, and reachability analysis to finite-state machine models of hardware. The latter class of systems work automatically but they do not always scale up efficiently to realistic hardware designs. The challenge then is to combine the generality of theorem proving with an acceptable and efficient level of automation.

  One way to retain generality without sacrificing automation is to integrate general theorem proving techniques with efficient and powerful decision procedures as well as finite-state enumeration procedures, such as model checkers. This combined approach to verification has formed the guiding design principle for the Prototype Verification System (PVS) developed at SRI. PVS combines a higher order logic with a rich type system to support an expressive specification language that is well-suited for a range of applications including the specification of hardware designs at different levels of representation. The primitive verification steps of PVS are implemented using an interacting set of automatic inference procedures. The inference procedures of PVS include a BDD-based boolean equivalence checker, symbolic simplifier (rewriting), and arithmetic and equality decision procedures. PVS provides the ability to invoke as a primitive inference step a model checker (based on Mu-calculus) for finite state systems.

  The ability to use powerful decision procedures and model checking as primitive inference steps within a higher order logic framework gives rise to a unique semi-automatic verification paradigm in which the tedious low level steps of verification are automated while the user need only provide guidance at a high level. For example, functional correctness properties of combinational circuits, finite-state sequential circuits,

and controller-datapath circuits, in which the controller is a finite state machine but the datapath state is arbitrarily large, can be verified automatically provided one has enough computing resources. In those cases where automatic verification is not possible due to resource limitations or because the design or the property being verified is too complex, one can use the logical framework to decompose or abstract the verification problem into a finite number of smaller parts each of which can be completed automatically.

The following lectures of the course will illustrate the use of the semi-automatic verification paradigm supported by PVS for hardware design verification. Although we will be using the PVS system for illustration, the methodology for hardware verification presented is applicable to any system that combines a higher order logic framework with decision procedures and finite state verification procedures. The PVS material will be presented by interspersing lectures with live demonstration of the PVS system on design examples.

- Overview of lectures:
  - An overview of the PVS verification system and its design philosophy: language, prover, and interface.
  - A general methodology for using a language based on higher order logic with types for specifying hardware designs at the gate level, register-transfer level and higher. We present both control-dominated as well datapath-dominated examples.
  - A verification methodology that uses symbolic execution (rewriting) and BDD-based boolean simplification for verifying datapath-dominated examples such as microprocessors. We illustrate the method on a small microprocessor but one that uses realistic design features, such as pipelining.
  - A verification methodology that uses induction proof techniques for verifying iterated hardware designs, such as bit-sliced ALUs and multipliers.
  - A verification methodology that combines theorem proving and model checking methods for verifying control-dominated examples, such as protocols.

Recent Publications in the BRICS Notes Series

NS-96-2 Allan Cheng, Kim G. Larsen, and Mogens Nielsen, editors. *Programme and Abstracts of the BRICS Autumn School on Verification* (Aarhus, Denmark, October 28 – November 1, 1996), August 1996. ii+18pp.

NS-96-1 André Berthiaume. *Quantum Computation. Mini-Course*. January 1996. iv+126 pp.

NS-95-6 Aravind Srinivasan. *The Role of Randomness in Computation*. November 1995. iv+99 pp.

NS-95-5 Robert Paige. *Analysis and Transformation of Set-Theoretic Languages. Mini-Course*. August 1995. iv+157 pp.

NS-95-4 Yuri Gurevich and Egon Börger. *Evolving Algebras. Mini-Course*. July 1995. iv+222 pp.

NS-95-3 Andrew D. Gordon. *Bisimilarity as a Theory of Functional Programming. Mini-Course*. July 1995. iv+59 pp.

NS-95-2 Uffe H. Engberg, Kim G. Larsen, and Arne Skou, editors. *Proceedings of the Workshop on Tools and Algorithms for The Construction and Analysis of Systems, TACAS* (Aarhus, Denmark, 19–20 May, 1995), May 1995. vi+334 pp.

NS-95-1 Igor Walukiewicz. *Notes on the Propositional μ-calculus: Completeness and Related Results*. February 1995. 54 pp.

NS-94-6 Uffe H. Engberg, Kim G. Larsen, and Peter D. Mosses, editors. *Proceedings of the 6th Nordic Workshop on Programming Theory* (Aarhus, Denmark, 17–19 October, 1994), December 1994. v+483pp.

NS-94-5 Andrew M. Pitts. *Some Notes on Inductive and Co-Inductive Techniques in the Semantics of Functional Programs,* DRAFT VERSION. December 1994. vi+135 pp.

NS-94-4 Peter D. Mosses, editor. *Abstracts of the 6th Nordic Workshop on PROGRAMMING THEORY* (Aarhus, Denmark, 17–19 October, 1994), October 1994. v+52 pp.

NS-94-3 Sven Skyum, editor. *Complexity Theory: Present and Future* (Aarhus, Denmark, 15–18 August, 1994), September 1994. v+213 pp.