Basic Research in Computer Science

Centre of
The Danish National Research Foundation

Computer Science Departments of
University of Aarhus and Aalborg University

BRICS, a Centre for Basic Research in Computer Science

BRICS, a Centre for Basic Research in Computer Science, is funded by the Danish National Research Foundation for the period 1994-1998. Its aim is to establish in Denmark important areas of basic research in the mathematical foundations of Computer Science, notably Algorithmics and Mathematical Logic. These are areas of great international significance. The Centre is to develop the areas as a joint effort between the theoretical Computer Science groups at University of Aarhus and Aalborg University, with dissemination to other Danish researchers as an important goal. The objective is to attract mainly foreign research experts, contributing to the research efforts of the Centre and to the training of young Danish researchers. The research plan is based on a commitment to develop Algorithmics and Logic integrated with existing strong activities in Semantics of Computation, using a combination of long-term efforts and a number of short-term, intensive programmes, within carefully chosen scientific themes. Organizationally, BRICS is an autonomous centre with its own management, and yet with its activities strongly integrated in the existing infrastructure and student environments at the two universities.

The central role of BRICS is to be a centre of expertise in the areas of Logic, Algorithmics and Semantics. Realisation of this aim will take advantage of involvement and contacts with researchers, projects and institutions internationally. BRICS will

- attract researchers of high calibre,
- produce academic materials, lecture notes; publications are to appear primarily in international journals and conference proceedings, but also in the Centre's own publication series,
- arrange/support summerschools and workshops, and distribute their proceedings,
- train senior students and researchers,
- foster expertise of potential relevance to practice.

The academic materials produced by the Centre will include some prototype systems produced by or in collaboration with the kernel researchers. It is expected that results produced by the Centre will chiefly be used by other researchers in universities and industry.

Although the Centre's concern will be within basic research, its results and expertise are of potential relevance to practice; formalisms, logical systems and algorithms developed, or their limitations understood, should contribute to future developments in high-level programming language design, mechanised checking and derivation of correctness of computer systems, including parallel systems, as well as their efficiency.

Research Areas

It is generally recognized that Algorithmics, Semantics and to a large extent Logic together constitute the backbone of Theoretical Computer Science, with Semantics and Logic dealing mostly with qualitative aspects and Algorithmics dealing mostly with quantitative aspects. Despite these differences in objectives there is an abundance of examples of problems in one discipline benefitting from results and techniques in another, and often stimulating research in another area. It is to be expected that their coexistence under a common centre will yield fruitful interaction between researchers and a productive cross-fertilisation between the different disciplines.

Logic

This century has seen Logic extend beyond the preserve of philosophers and philosophically-minded mathematicians, to become a central tool in understanding and reasoning about computer systems. The original motivation for formal systems, to make the activity of mathematics an object of study in itself, has arguably been supplanted in importance by the everyday use of logical systems in Computer Science. Logic touches on many aspects of computing. Its use ranges from that in the execution of machines, as in Logic Programming, or in the machine assistance to reasoning of theorem provers, to argumentation within a specialised program logic to verify that a program or system does what is required of it. Unquestionably, Computer Scientists are important users of Mathematical Logic, with problems in Computer Science giving new impetus to Logic.

A variety of logics are widespread in Computer Science. The familiar Predicate Calculus finds applications in a range of diverse areas from Program Correctness, Logic Programming, Database Theory, to knowledge representation in Artificial Intelligence. However, the role of Logic in Computer Science reaches far beyond that of classical Predicate Calculus. Notable has been the upsurge in uses of Modal and Temporal Logic in reasoning within areas as far apart as parallel programs and data-security. Historically, Constructive Logic arose out of philosophical objections some mathematicians had to too free-wheeling a treatment of infinite objects. In Constructive Logic the meaning of a proposition consists, roughly speaking, of its proofs, and is not taken to be simply true or false. Its constructive character is shared by computing systems, making constructive logics ideal in many applications. The demands of Computer Science are not always well-met by traditional Logic. The relatively young field of Categorical Logic is providing a suitably rich perspective in which to explore new developments in the Semantics of Computation, Program Logics and Type Theories, and through them the possibilities for new programming languages and methodologies.

Computers bring Logic to life; the numerous, tedious formal manipulations required by proofs within a formal system can be made actual only through the use of machine power. Precisely how to do this efficiently is a subject of intensive study involving several areas. The need for human interaction is recognised in most theorem-proving systems, often allowing the users to program strategies for proof through the medium of a programming language. Occasionally though, as in model checking, type inference or term rewriting, specific problems can be fully-automated.

Logic presents a guiding influence on the methodology and

design of computing systems. At the same time, trends in Computer Science help mould the development of Logic. The challenge of putting Logic to work has not only led computer scientists to new variants of the logics and to efficient techniques for establishing logical properties, but also uncovered new areas for logical and mathematical investigation.

Algorithmics

Algorithmics—the design and analysis of algorithms and Complexity Theory—constitutes the scientific foundation for reasoning about quantitative aspects of computing, i.e. the use of resources like computing time and storage. Examples of problem areas are: robotics, geometrical objects, pattern matching, transport and communication systems, graphs, logical theories and proof systems.

Design and analysis of algorithms is concerned partly with the design and analysis of efficient algorithms solving concrete problems, partly with identifying common patterns of problems and associated algorithmic paradigms leading to efficient solutions. Traditionally, research has focussed on deterministic sequential algorithms, but non-sequential and randomized algorithms are of increasing interest. As an orthogonal issue, data-structures play an important role, partly because every algorithm needs to organize data systematically, partly as common grounds for the development of efficient and user-friendly algorithmic libraries.

The goal of Complexity Theory is to determine the amount of computational resources needed to perform certain computational tasks. The efforts to achieve these goals have only been partly successful, the major shortcoming being the inability to prove lower bounds for general models of sequential computation. One trend is therefore to study other models (including models for parallel computation). Through characterization of various computational models according to their relative computing power it has often been possible to give quite accurate estimates for how difficult a computational task is.

Even though Complexity Theory is a theoretical discipline, it has had important practical applications, e.g. in the area of Cryptography, where most modern techniques are based on the fact that large instances of problems belonging to hard complexity classes, really are unsolvable in practice.

The relationship between Semantics, Logic and Algorithmics is not limited to the need for efficient algorithms for logics and program analysis. Algorithmic problems in Logic and Semantics have also played an important role in the development of algorithmic techniques. Classical examples are algorithms for formal languages and automata, the need for which arises in connection with efficient language processors (compilers and interpreters). Another example is the development of dynamic algorithms to support on-line applications, where incremental language processors represent an interesting source of inspiration.

Apart from being a source of inspiration, Semantics and Logic have a role to play in the development of algorithms and complexity. As an example, it is possible to describe fairly abstractly the classes of combinatorial problems that are solvable by way of some of the fundamental algorithmic paradigms, such as divide and conquer and dynamic programming. There is reason to hope that such descriptions could be extended to some more recent paradigms in randomized, parallel and dynamic algorithms, thus increasing insight in Algorithmics by way of concepts and techniques from Semantics.

An interesting topic on the borderline between Algorithmics and Logic is the question of alternative characterisations of important complexity classes. There are such results for several complexity classes and it would be of considerable interest to extend this work to other classes. A related example is the connection between nondeterministic algorithms and proof checking in formal proof systems.

Semantics

The purpose of Semantics of Computation is to give programming systems and languages a mathematical meaning, which is at the same time abstract (i.e., ignores as much irrelevant detail as possible) and built-up compositionally from the structure of the system or program.

There are several important applications of such formalizations: they permit formal analysis and verification of programs and systems before use, avoiding expensive or disastrous errors in practice; they serve as guidelines for programmers and language implementations; in some cases it is even possible to generate a language implementation systematically and/or automatically from its semantic description.

Historically, Semantics has focussed on qualitative aspects of programs, and its scientific development has followed closely traditions from Mathematical Logic, e.g. in model-based denotational semantics, and in deduction-oriented operational semantics. The unsettled area of the semantics of parallel (or concurrent) computation is often founded on an operational semantics, while denotational semantics, broadly successful for sequential languages, has its original mathematical foundation (Domain Theory) in certain kinds of topological spaces. Subsequently, the desire to have denotational semantics which agree more closely with operational semantics has led to new domain theories. Developments in Constructive and Categorical Logic are yielding a refined analysis of what is required of a domain theory and the logical principles that hold of it. Presently the modelling and analysis of parallel computation and traditional denotational semantics inhabit rather separate worlds. Ultimately we cannot expect the theory of parallel computation to remain so separate a study. Categorical Logic furnishes a promising tool to see its place in the general setting of a theory of computation.

Various logical formalisms for reasoning about programs and systems have been constructed around semantics. These range from specific program logics, to general logics of computable functions, to type systems expressing program properties. Modal and temporal logics are important tools in specification and reasoning, especially for parallel languages.

Of course, like most areas of Computer Science, Semantics and the analysis of programs often have call on results, techniques and concepts from Algorithmics. Model checking and type checking, for instance, draw heavily on such expertise. This tendency can only increase as Semantics extends to treat more quantitative aspects of programming systems.

Activities

Research Themes

In addition to the longer term research activities, it is intended to concentrate on research themes. One theme is planned per year, keeping close to the university semester structure when possible. Guests, seminars, lectures and workshops will be focussed around these activities. An aim is that each programme include a set of lecture courses lasting around two weeks, suitable both for graduate students and more senior researchers. Whenever appropriate these will be announced widely, for example, as summerschools.

Positions and Guests

In addition to the kernel researchers, at any one time BRICS is expected to host between 10 and 15 further researchers and guests. Information on these positions is being announced internationally.

PhD Training

The education of young researchers is an important objective of BRICS. At Aarhus, in particular, the Centre is expected to form an important part of the newly established graduate school. Several PhD studentships are to be announced each year.

Workshops, Summerschools and Conferences

It is planned to organize a number of Workshops, Summerschools and conferences.

Dissemination

Since the dissemination of the results of our effort into the Danish research system is an important measure of success, the Centre will seek collaboration with a wide spectrum of research institutions, involving a number of open arrangements, like summerschools, PhD courses, and seminars. The information on such activities and plans for future programmes will be made widely available through (electronic) newsletters from the Centre. Research and course material from the Centre will be made publicly available through the Centre's lecture-notes and publication series. For further information and enrolment to the BRICS newsgroup contact the BRICS addresses below.

Applications

Applications for positions should preferably be sent by e-mail and include curriculum vitae and three names of referees for recommendations as well as the referees' regular mail addresses and, if possible, e-mail addresses.

Kernel Researchers

Director
Glynn Winskel, Professor (Aarhus)
Co-Directors
Mogens Nielsen, Associate Professor (Aarhus)
Erik Meineche Schmidt, Associate Professor (Aarhus)
Project Manager
Uffe H. Engberg, (Aarhus)

Kim Guldstrand Larsen, Professor (Aalborg)
Peter D. Mosses, Associate Professor (Aarhus)
Michael I. Schwartzbach, Associate Professor (Aarhus)
Arne Skou, Associate Professor (Aalborg)
Sven Skyum, Associate Professor, Reader (Aarhus)

Addresses

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
Department of Mathematics and Computer Science
Aalborg University
Fredrik Bajers Vej 7
DK - 9220 Aalborg Ø
Denmark

Telephone: +45 9815 4211 # 5040
Telefax: +45 9815 8129
Internet: BRICS@cs.auc.dk

Further Information

Further information on BRICS can be accessed through World Wide Web (WWW) and anonymous FTP. To connect to the BRICS WWW entry, open the URL:

```
http://www.brics.dk/
```

The BRICS WWW entry contains updated information about activities, courses and researchers as well as access to electronic copies of information material and reports of the BRICS Series (look under Publications).

To access information material and reports of the BRICS Series via anonymous FTP do the following:

```
ftp ftp.brics.dk
cd pub/BRICS
get README.
```