Six issues concerning future directions in concurrency research

ACM Computing Surveys **28**(4es), December 1996, http://www.acm.org/pubs/citations/journals/surveys/1996-28-4es/a39-baeten/. Copyright © 1996 by the Association for Computing Machinery, Inc. See the permissions statement below.

Six Issues Concerning Future Directions in Concurrency Research

Jos Baeten
Computing Science Department, Eindhoven University of Technology
P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands
josb@win.tue.nl, http://www.win.tue.nl/win/cs/fm/josb/

Jan A. Bergstra
Faculty of Mathematics & Computer Science, University of Amsterdam
Kruislaan 403, NL-1098 SJ Amsterdam, The Netherlands
janb@wins.uva.nl, http://www.wins.uva.nl/research/prog/people/janb/

Important future research directions are the combination of processes and data and the elaboration of non-functional requirements. Telecommunication will remain an important application area. The application area of embedded software will become increasingly important. Further, research into testing theory will become important. In the area of tool support, further development of theorem provers and checkers and integration of theorem proving and model checking deserves attention. Last, embedding of concurrency theory into an encompassing (object-oriented) design method is worthwhile.

1. Processes and Data

An important direction for fundamental investigation is the connection between datatypes and process algebra. The starting point is diverse: specification languages like LOTOS and PSF combine algebraically specified datatypes with process algebra syntax by having processes, actions and conditional constructs depending on data parameters. In these languages data are imported at a high level of abstraction. Pi-calculus, on the other hand, allows an embedding of lambda-calculus. Owing to that, it forms an extension of a process algebra to a calculus in which data manipulation of arbitrary complexity can be represented. Pi-calculus includes low-level but universal primitives for data manipulation. Both styles deserve further research.

Our agenda, in particular, is to proceed along the first line (LOTOS, PSF) and to investigate aspects like error values and undefined objects in the datatypes, and of course modular structure of data in relation to the various process algebras. As relevant work we mention a paper in FAC 1994 in which we survey data input mechanisms in ACP, CCS and CSP and a recent report that provides a data-type specification format involving an error value and an undefined value that may serve as a starting point in research about the role of error and undefined data values in a process-algebra setting.

2. Non-Functional Requirements

The focus in the application of concurrency theory has always been on the specification and verification of functional requirements. More and more, non-functional requirements such as timeliness, fault tolerance, availability, security, and safety are also becoming important, especially in the construction of embedded systems. The addition of timing information to process algebra has received much attention in recent years, and an increase in the number of applications has shown that a certain level of maturity has been reached. The situation with the addition of probabilistic and stochastic information is much less advanced. A number of studies have appeared, but no stabilization or significant application has yet occurred. The combination of timing and probabilistic theories and generalization to stochastic process theory will require research in the coming years.

3. Embedded Systems

Embedded systems are computer systems that form an integral part of a larger system like a production cell, a controller for a home heating system, an audio or television set, a copying machine, an aeroplane, a digital telephone exchange etc. The term ``embedded system'' thus encompasses a broad class of systems, ranging from simple microcontrollers to large and complex multiprocessor and distributed systems. Embedded systems control industrial or physical processes. Sensors continuously gather information from the environment. The service of the embedded system is to process this information and to signal the actuators in accordance with the mechanisms of the controlled process. Inherently, timing information plays a crucial role in the functioning of an embedded system. Often, probabilistic information also plays a role, for instance when considering failure rates and fault tolerance.

The construction of embedded systems is not standardized, nor is it supported by standard software packages. It is estimated that the number of people constructing embedded systems will increase rapidly over the following years, much more than the number of people involved in the construction of administrative information systems.

4. Testing

Software testing is highly relevant because in the industrial practice of software development it often takes a significant fraction of the total development effort. For testing protocols, ISO has established a framework containing terminology and concepts, based on the language TTCN. Important issues in protocol testing are the distinction between conformance testing and interoperability testing, and the multi-vendor nature of the telecommunication industry. Not only is software to be tested, but also combined hardware-software systems. Often, communication systems are specified and realized using SDL or a process algebra such as LOTOS or PSF in combination with C. The distance from these high-level languages to TTCN is quite large, which may hinder automated test generation. Use of process theory may help bridge the gap.

5. Automated Verification

One succesful and even commercially used approach is model checking, which makes it possible to check the correctness of programs exhaustively without any user interaction. However, in principle as well as in practice, there are limitations on the state space of the programs.

Another approach, theorem proving, avoids these limitations by manipulating formulas rather than searching a state space, but has other drawbacks like difficult user interfaces and the need for user participation. A challenging but promising route is therefore to combine model-checking and theorem-proving techniques. Viewed from the proof-generator/checker end, proof strategies need to aim for a reduction to subproofs that can be relegated to the model checker. Viewed from the model-checking end, how can those parts of a problem that require proof generators/checkers be separated out? It is interesting to investigate integration of the two approaches in one tool, in which the switch from theorem proving to model checking and back again can be made autonomously, without user intervention.

6. Design Method

Formal methods are used most effectively only in critical parts of a system, and only in a part of the system design process. Therefore, it is desirable to integrate (or at least, interface) a formal method in an encompassing design method. This will also help acceptation and adoption by industry.