PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1988,
edited by ICOT. © ICOT, 1988

The Panel on Theory and Practice of Concurrent Systems

Ehud Shapiro
The Weizmann Institute of Science
Rehovot 76100, Israel

September 14, 1988

The panel on theory and practice of concurrent systems brings together researchers who approach the problems of concurrency from radically different angles. It is my hope that by exposing and confronting the different perspectives of the penalists all of us will enhance our understanding of what are the problems, concepts, current research directions, and the long term visions of this area of investigation.

In this short note I will attempt to outline the perspectives represented by the penalist. This presentation does not attempt to convey the present ideas and positions of the penalists; these will be presented in the accompanying position papers, written concurrently with this note. Rather, it represents my subjective impressions of the research directions pursued by the panelists. The description is necessarily rough and terse, and since it was not screened by the other panelists, may contain impressions which are either wrong or not up to date. Specifically, the panelists may present a position different than the one ascribed to them in this note.

William Dally represents a research direction that begins with architectural considerations and ends in system and language design. Like the dataflow people a decade ago, hardware architects such as Dally and Chuck Seitz find computational models based on fine grain concurrency to be the best match for parallel computer architectures. At the time, dataflow research faced an immature hardware technology and hardware development tools and, more importantly, the lack of abstract computational models and programming languages suitable for their purposes. This required the development of dataflow languages based on single-assignment (or write-once) variables.

Presently, there are an abundance of models and languages to choose from. The differences between the object-oriented models Dally and colleagues are investigating and early dataflow models is primarily the grain-size: each unit of sequential execution consists of tens or hundreds of the equivalent of conventional machines instructions, compared to one or few instructions in early dataflow models. This makes practical the use of the well-understood and highly optimized von Newman processor as the building block of the concurrent computer. The research effort is invested mainly in the interconnection network, the communication protocols, resource management, and efficient message handling.

Goeffrey Fox represents the practitioners of concurrent computing. While many are still contemplating whether there is enough concurrency in real-life problems to justify large scale concurrent computers, Fox and his colleagues, working closely with Seitz since the days of the Cosmic Cube, have demonstrated that many computational problems encountered in Physics and Chemistry are amenable to efficient solution on concurrent computers. They have found many "embarrassingly parallel" problems, that is problems amenable to concurrent solution with almost no intellectual effort, and with relatively little implementation efforts. They have also found many other problems that are amenable to efficient parallel implementation using more sophisticated algorithms. They have demonstrated, contrary to what many computer scientists haunted by FORTRAN want to believe, that FORTRAN augmented with *send* and *receive* primitives can go a long way on non-shared memory concurrent computers.

Carl Hewitt represents the object-oriented approach to concurrent systems. This approach suggests that a concurrent system be structured as a collection of communicating objects. The basic operations of an object are receiving and sending messages, changing state, and creating new objects.

Although the object-oriented approach to concurrency was quite radical when first suggested, its ideas have by now penetrated almost all other approaches to concurrency, including the architecture-oriented approach to concurrency mentioned above, the theoretical message-passing models of concurrency, as well as the concurrent logic programming approach, mentioned below.

Related to the object-oriented approach are open systems concerns. In a world of ever-expanding computer communication networks, the concept of a computer system being a closed entity with a fixed set of entities to interact with is no longer valid. Computational objects may join a network, cease to exist, or even change their protocols of interaction dynamically. The goals of research in open systems is to devise techniques and languages that can be used to specify computational objects and systems that can survive in such a dynamic open world. It is stipulated that the object-oriented approach to concurrency may offer a foundation for such open systems.

Robin Milner represents the approach of studying concurrency via abstract calculi such as CCS, CSP, temporal logic, and, more recently, UNITY. This approach devises mathematical models of concurrency and studies them with the goal of increasing our understanding of the fundamental properties and problems of concurrent systems.

Within an abstract setting it is easier to address questions such as program equivalence, compositionality and equivalence of program parts, and fairness, as well as the superposition of algorithms for detecting properties of process networks.

Sometimes, as in the case of CSP and OCCAM, an abstract model gives rise to a concrete programming language that preserves many of its ancestor's properties. However, often there is an undesirable gap between the theoretical and experimental investigations of concurrency. One example is research in semantics. The dominant theoretical message-passing models are synchronous, whereas the majority of concrete models are asynchronous (including object-oriented languages, logic languages, and FORTRAN + send/receive, excluding OCCAM). Another example is research in complexity, where the dominant model used in studies of parallel complexity and parallel algorithms is the synchronous shared memory PRAM. However, most successful experimental work on parallel algorithm was carried out on non-shared-memory asynchronous parallel computers.

It is my hope that the interdisciplinary nature of this conference in general and of this panel in particular will help to bridge this gap between the theory and practice of concurrent systems.

Kazunori Ueda represents the "middle-out" approach to the study of concurrent systems, taken by ICOT and related research groups. In this approach an abstract computational model, with associated programming languages, serve as the starting point for both top-down and bottom-up investigations. The goal of these investigations is the construction of a comprehensive parallel computer system based on this model. ICOT has chosen the concurrent logic programming model and the languages GHC and FGHC as the basis for their investigations.

In the top-down investigation implementation questions are considered: what are suitable architectures for the computational model, and how to implement the language efficiently on target architectures. In the bottom-up investigations the use of the languages, as well as its properties, are investigated. Useful programming techniques are identified, methods for implementing both system programs and application programs in the language are studied, and questions of program developments, program analysis and transformation, including semantics and program equivalence, are pursued.

The integrity of such a broad-spectrums investigation is maintained by adhering to the principle that the abstract computational model is a strict layer of abstraction. This layer of abstraction serves as the platform from which both the top-down and the bottom-up investigations begin, and a meeting point for those who maps the abstract computational model on a concrete architecture, and for those who use the computational model.

David H.D. Warren represents the research aimed at harnessing concurrent computers by parallelizing "conventional" languages. Such research aims at providing better cost/performance ratios by exploiting concurrency without changing language semantics. This research efforts were carried out for FORTRAN and more recently by Warren and colleagues, for Prolog. The philosophy behind this research direction is that concurrent languages (i.e. languages that can express concurrency) are harder to use, and don't have a large software base. Consequently, one should offer programmers a language they know and like (e.g. FORTRAN or Prolog) which does not contain explicit constructs for expressing and controlling concurrency (and in this sense may be higher-level). The task of mapping such a language effectively on a concurrent computer resides with the compiler and runtime system, and not with the programmer.

Such research is often torn between two conflicting goals: one is to provide the programmer with a high-level notation that can be parallelized effectively. The other is to preserve the original, sequential, semantics of a known programming language, which often has constructs that hinder parallel execution without having other clear benefits.

This dilemma suggests defining novel high-level languages that still shelter the programmer from the complexities of concurrency, but differ from their sequential ancestors in being "cleaner" and better amenable to parallel execution. An example of such a language is Andorra, developed by Warren and colleagues.