Time for Time ...

Ernest Allen Emerson II

Computer Sciences Department
University of Texas at Austin

Amir Pnueli Memorial,

New York, NY,
8 May 2010

Ultimate Goal of FM: To Program Well

- **Basic Need**: **predictable & reliable** programs
- **Program**:: hardware design, software program, system, etc.
- **Problem**: programs have **bugs**
- **Issue**: **Programs** are Mathematical Objects
- **Solution**: **Formal Methods** based on Mathematical Logic
- **Specify**: **correct** behavior
- **Verify**: program **conforms** specification

Amir Pnueli (1941 – 2009)

* father: professor of Hebrew literature

* Ph.D. dissertation at Weizmann Institue:
- Solution of Tidal Problems
- in Simple Basins, 1967 (advisor: Pekeris)

* postdoc: Stanford w/ McCarthy

* seminal paper [Pnueli 77] while visiting Penn
- *Logic of Commmands* suggested by Saul Gorn; blurb on back:
- Rescher & Urquhart, *Temporal Logic*

* Newton of Temporal Logic
- Tarski of Computer Aided Verification

Bumping into Amir

Lop81, Popl83, Lop83, Monterrey84, Stoc84?, Icalp84?, Popl85, Lop85, Lics86, UT-Fall86, Manchester87, Popl89...

Comments

“Amir Pnueli plainly deserves the Turing Award”
— Krzysztof Apt, $\approx$ 1987

“Pnueli is the single scientist I most admire and respect professionally.”
— Emerson to Dijkstra, 1994
— 3 hr discusion
— Dijkstra appreciates Pnueli’s excellence

Verification Engineering: A Future Profession

Amir Pnueli
Weizmann Institute of Sciences

An A.M. Turing Award Lecture
PODC, San Diego, 23.8.97

Formal Verification

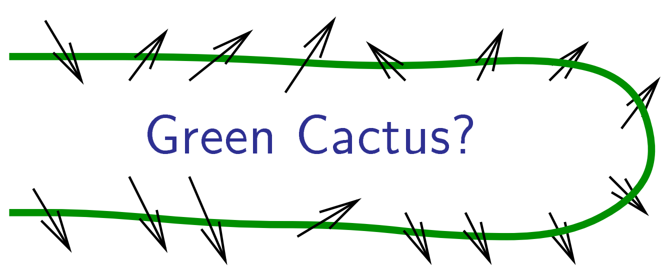
Started with sequential program verification which, so far, has not been universally embraced.

It then expanded into the are of reactive system verification, where it has a more visible impact and greater success. Why?

Distinguish between [HP85]

- Transformational systems (sequential): Run in order to produce a final result on termination. Can be modeled as a black box. Specified in terms of their Input/Output relations.

- Reactive systems, whose role is to maintain an ongoing interaction with their environment.



Such systems must be specified and verified in terms of their behaviors.

Originally,

Formal verification was associated with the application of axiomatic or deductive techniques to proofs of correctness.

Things having to do with logic.

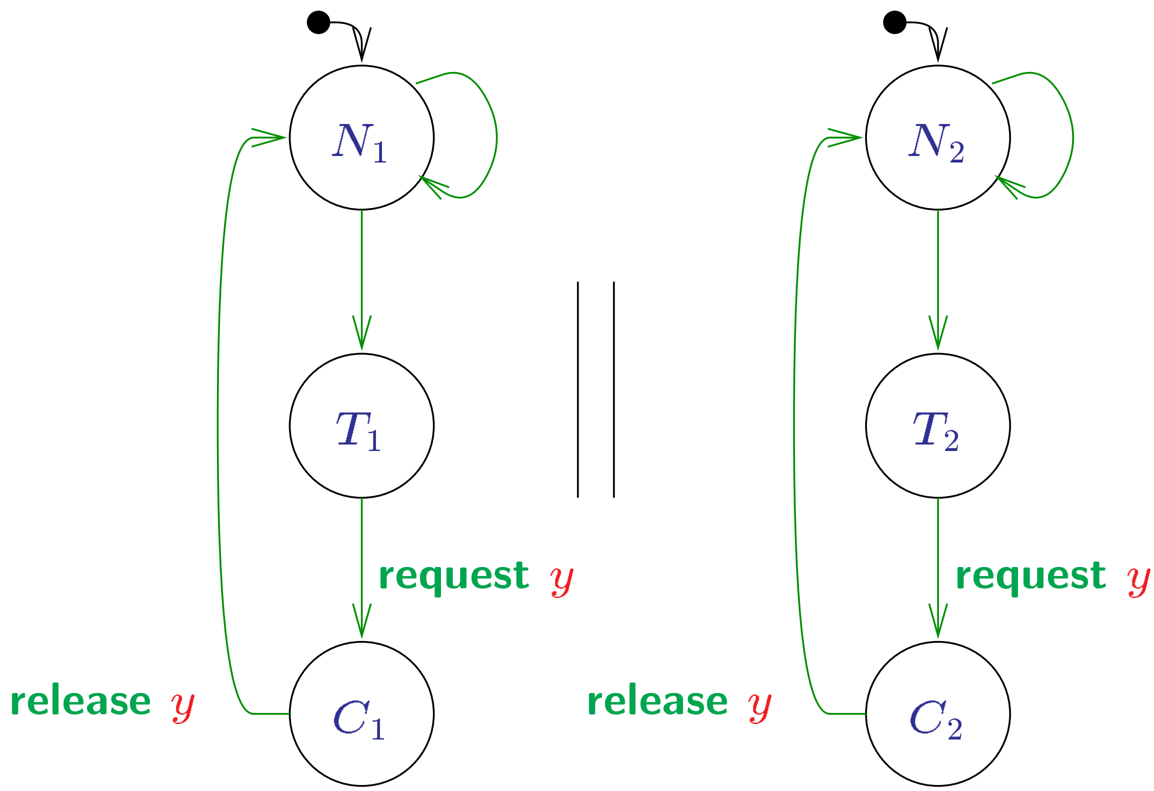
Since the early 80's [CE81], it also includes model-checking and other algorithmic approaches, which can be viewed as exhaustive simulation or exhaustive testing.

A first step towards engineerization of the field!

Example: Mutual Exclusion by Semaphores

Two processes coordinating access to their critical sections by Semaphores —

y: **integer where** $y = 1$



The semaphore instructions **request** y and **release** y stand for

$$\langle \textbf{await } y > 0\,;\; y := y - 1\rangle \quad \text{and} \quad y := y + 1.$$

Specification of MUTEX by a Property List

- Safety:

$$\Box \neg(C_1 \wedge C_2)$$

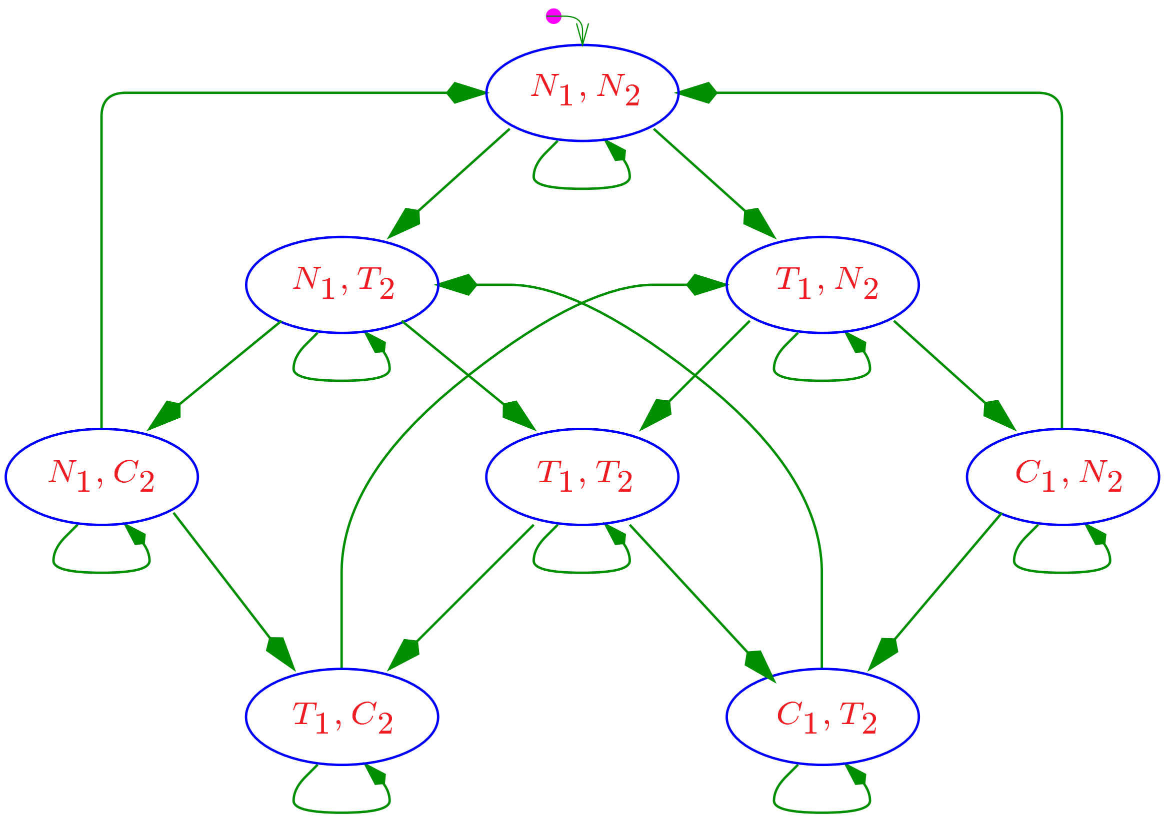
  The two processes can never visit their respective critical sections at the same time.

- Liveness:

$$T_1 \Longrightarrow \Diamond C_1 \qquad T_2 \Longrightarrow \Diamond C_2$$

  Every visit of a process to its trying section is followed by a visit to the critical section of the same process.

Specification by an Abstract Model



The absence of the state $\langle C_1, C_2 \rangle$ implies mutual exclusion.

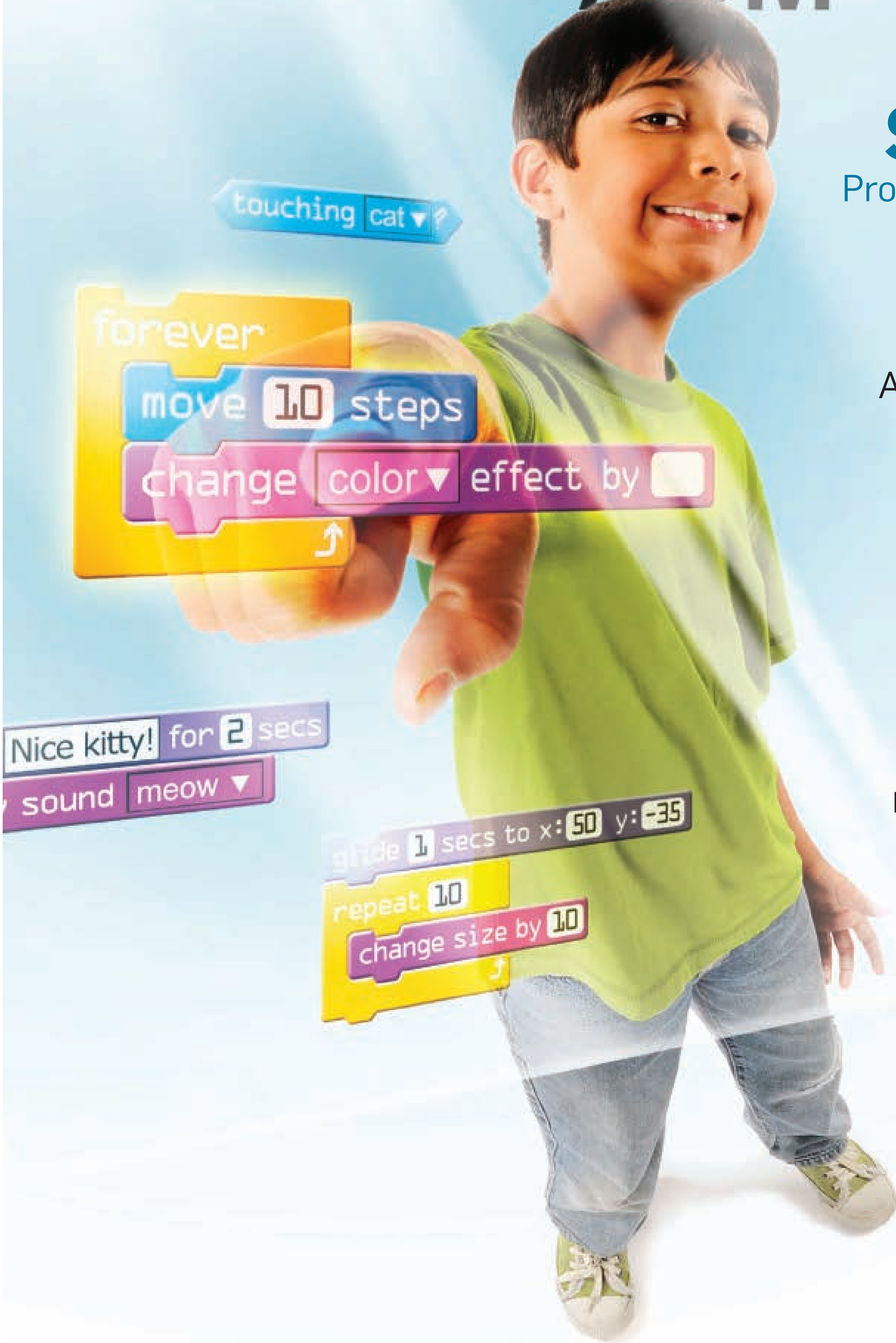
Personal

Pnueli's Turing Award Lecture, 1997
— Cites two papers
— [HP85] Reactive systems
— [CE81] Model Checking
— uses Mutex example of [EL85] (cf. [CE81])
— I felt very honored

COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

11/09 VOL.52 NO.11



Scratch

Programming for All

Communications Surveillance

An Interview with Ping Fu

Usable Security: How To Get It

E-Paper's Next Chapter

Turing Lecture

by Edmund M. Clarke, E. Allen Emerson, and Joseph Sifakis

Association for Computing Machinery

11/2009
VOL. 52 NO. 11

Practice

Article development led by acmqueue
queue.acm.org

Contributed Articles

Review Articles

Research Highlights

About the Cover: As if they were assembling Lego bricks, children snap together Scratch graphical programming blocks—shaped to fit together only in ways that make syntactic sense—to create their own programs, playfully explored in the cover story beginning on page 60.

Virtual Extension

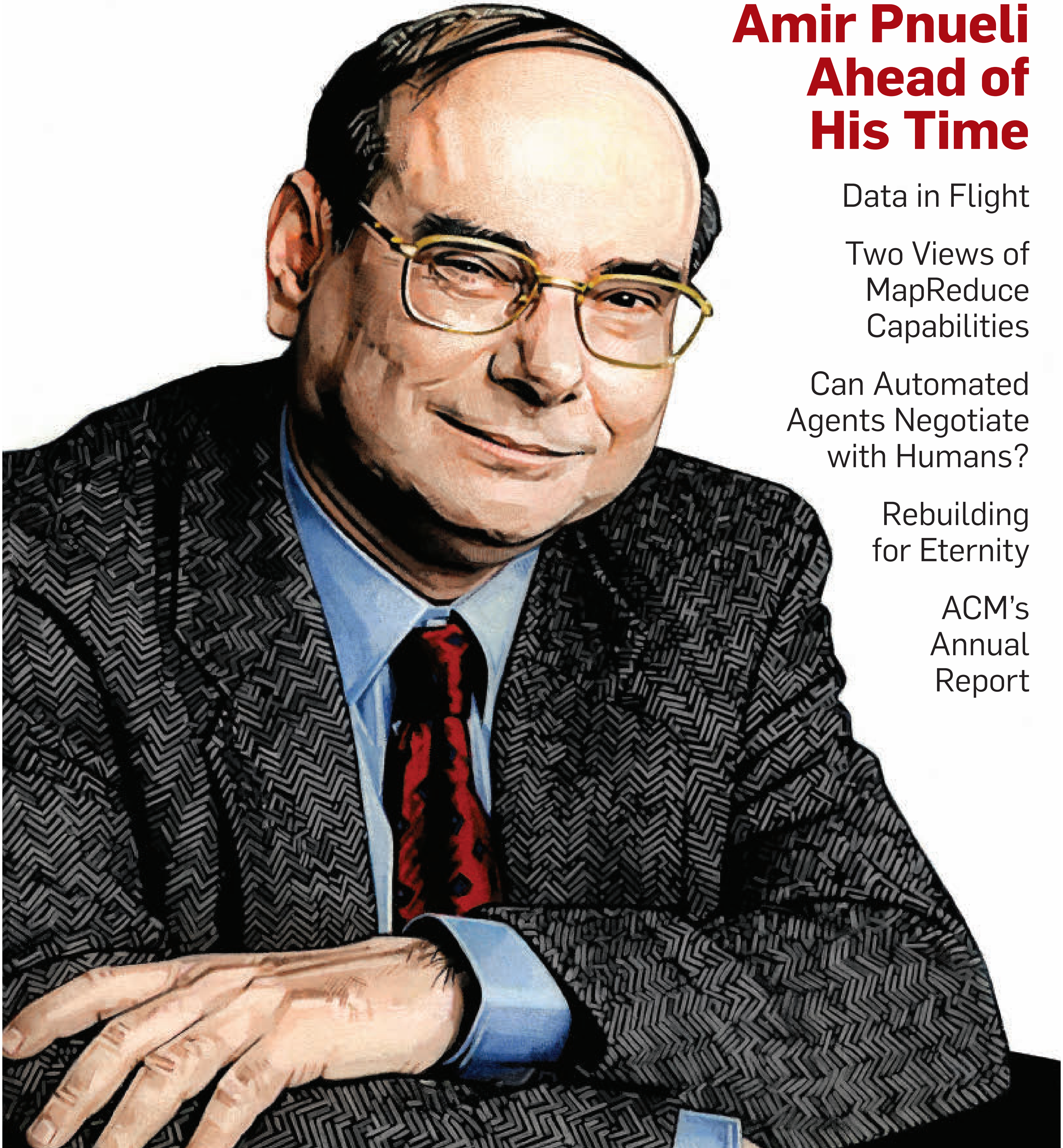
As with all magazines, page limitations often prevent the publication of articles that might otherwise be included in the print edition. To ensure timely publication, ACM created *Communications*' Virtual Extension (VE).

VE articles undergo the same rigorous review process as those in the print edition and are accepted for publication on their merit. These articles are now available to ACM members in the Digital Library.

COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

01/2010 VOL.53 NO.01



Amir Pnueli Ahead of His Time

Data in Flight

Two Views of MapReduce Capabilities

Can Automated Agents Negotiate with Humans?

Rebuilding for Eternity

ACM's Annual Report

DOI:10.1145/1629175.1629176

Moshe Y. Vardi

More Debate, Please!

In the May 1979 issue of *Communications*, a powerfully written article by Richard A. De Millo, Richard J. Lipton, and Alan J. Perlis entitled "Social Processes and Proofs of Theorems and Programs," argued that formal verification of programs is "difficult to justify and manage." The article created the perception, in the minds of many computer scientists, that formal verification is a futile area of computing research.

That article did not cite a 1977 paper by Amir Pnueli entitled "The Temporal Logic of Programs." His paper had attracted little attention by 1979, but by 1997 it would be described as a "landmark paper" in the citation that accompanied Pnueli's 1996 ACM A.M. Turing Award. In his paper, Pnueli, whose sudden and unexpected death on Nov. 2, 2009 shocked the computer science community, laid the foundation for formal verification of concurrent and reactive programs. (An article describing Pnueli's scientific legacy appears on page 22.) The paper also laid the foundation for the development of model checking, an automated formal-verification technique for which Edmund A. Clarke, E. Allen Emerson, and Joseph Sifakis received the 2007 ACM Turing Award.

With hindsight of 30 years, it seems that De Millo, Lipton, and Perlis' article has proven to be rather misguided. In fact, it is interesting to read it now and see how arguments that seemed so compelling in 1979 seem so off the mark today. Should we infer that *Communications* erred in publishing that article? My answer is a resounding "no!"

My basic education included exposure to Talmudic scholarship. Jewish scholars in the first half of the first millennium believed that truth will emerge from vigorous debate. The Talmud, a monumental work of Jewish scholarship concluded circa 500 CE, is in essence a compendium of legal debates. Vigorous debate, I believe, exposes all sides of an issue—their strengths and weaknesses. It helps us to reach more knowledgable conclusions. To quote Benjamin Franklin: "When Truth and Error have fair Play, the former is always an overmatch for the latter." In my opinion, however, the editors of *Communications* in 1979 did err in publishing an article that can fairly be described as tendentious without publishing a counterpoint article in the same issue. Indeed, the article instigated so many reader responses, the editors published 10 pages of letters in the November 1979 Forum section of *Communications*, calling the work everything from "marvelous" to "humorous."

In 2007, when I met with various focus groups to discuss the relaunching of *Communications*, I was encouraged to keep this publication engaged in controversial topics. "Let blood spill over the pages of *Communications*," said one discussant jokingly. At the same time, however, participants believed that the magazine should represent all points of view fairly. This sentiment led to the establishment of the Point-Counterpoint feature, in which both sides of an issue are represented by opposing articles. Quoting Franklin again: "when Men differ in Opinion, both Sides ought equally to have the Advantage of being heard by the Publick."

Since the relaunch in July 2008, we have published several Point-Counterpoint pairs: on computing curricula, e-voting, Net neutrality, and the direction of CS education in the U.S. At this point, however, the pipeline for such articles is dry. I had assumed that both members of the editorial board and readers would propose topics for Point-Counterpoint articles, but that does not seem to be the case. It is almost as if people believe there is something improper about engaging in direct debate. In fact, several authors whom I invited to participate in Point-Counterpoint debates have declined in order to avoid head-on confrontation. The truth is, however, that there are many issues in computing that inspire differing opinions. We would be better off highlighting the differences rather than pretending they do not exist.

In this issue of *Communications* we have a debate that is quite a rarity in computing research: a *technical* debate. MapReduce (MR) is a software framework to support distributed computing on large data sets on computer clusters. It was introduced by J. Dean and S. Ghemawat of Google in a highly influential 2004 article, and featured as a Research Highlight paper in the January 2008 issue of *Communications*. The success of MapReduce led some to claim that the extreme scalability of MR will "relegate relational database management systems (RDBMS) to the status of legacy technology." A pair of Contributed Articles in this issue—Dean and Ghemwat on one side and Stonebraker et al. on the other—debate the relative merits of MR and RDBMS beginning on page 64. As parallel computation is one of the hottest topics in computing today, I have no doubt that our readers will find this technical debate highly instructive.

If you have topics that you think should be debated on the pages of *Communications*, please contact me. More debate, please!

Moshe Y. Vardi, EDITOR-IN-CHIEF

Impact of Amir Pnueli

— **Specification – temporal logic**: seminal [Pn77] onward

— **Ongoing behavior** recognized as important, practical

— **Verification, deductive**: 1977 ownward

— **Verification, algorithmic**: fundamental [LP85] onward

— **Synthesis, algorithmic**: 1989 influential [PR89] onward

— **Games**: solving using (vectored) mu-calculus
...

Temporal Logic per se and Its Origins

* **a form of modal logic:**

– developed by philosophers

– $\Box p$ necessarily p: Gp always p

– $\Diamond$ possibly p: Fp sometime p

* **Prior 67 credited w/ invention**

– speculated on use for

– describing workings of digital computers

– Prior working in 50's, 57 book

* **Prior credits teacher Findlay**

* **Philosophers argue goes back to**

– Medieval Logicians

– Ancient Logicians

* Ohrstrom & Hasle,

"Prior's Re-discovery of Temporal Logic"

Other Efforts

* Pnueli cites Burstall 74, Kroger 76 ...

* These and other efforts to formulate and use
- Modal, Tense, Dynamic, etc. logics in CS
- were interesting and valuable

* But had little impact
- over the long term
- and upon practice

* Pratt vs Pnueli debate in 81:
- Pratt – Dynamic Logic subsumes TL
- Pnueli – TL will win based on pragmatics

Isaac Newton Founded Calculus

* Newton invented (or founded) calculus

* Newton applied it to solve most basic questions
- in physical science
- provided **Profound Revolution** in physical science

* Newton built on prior work
- of other mathematicians, studying curves
- Isaac Barrow: slope
- Archimedes: area

* Liebniz also discovered calculus
- more useful notation

Amir Pnueli
Founded Temporal Logic

* Pnueli invented (or founded) temporal logic

* Applied it toward solving most basic questions
- in computer science
- **Paradigm Shift** in Formal Verification

* Pnueli built on prior work
- major impact on applications
- major advances in temporal logic too
TL elegant: notation, notation, notation
- tailored, succinct: $\forall$, $\exists$, F,G,X,U

Pnueli Founding
TL in CS

* Founded temporal logic in CS

* Guided and Developed it !!!

* Why Pnueli 77 so Seminal?

- Pnueli emphasized importance of infinite behavior

- Examples: operating systems

- Specification is essential, more fundamental than verification

- Temporal logic is very natural for specification

- "Sometimes", "always" easy to use

- Gave natural proofs of e.g. mutex

- Captured the imagination just as Hoare 69

Just a Tiny Fraction of Amir's Work

* He published 250+ papers

* He worked on, pioneered, and foreshadowed many different topics

- abstraction
- past tense
- automata
- parameterized systems
- language containment paradigm
- algorithmic reasoning
- deductive reasoning
- automata-theoretic approach

Future? TL + Automata?

* TL formulas **are** automata [Em94]

* Automata can be advantageous
- Uniform framework: modelling, spec'n, ver'n, synth.

* Background: Tactics
- [St81] automata-theoretic SAT pgm logics
- [ES83],[WVS83] early "compilation theorems"
- [Va85] Tames "automata-theoretic methods" of [St81]
- [LP85] LTL algorithmic ver'n using tableaux

* Important Strategy
- [VW86] Automata-theoretic LTL model checking
- exp. time worst case, often efficient in practice
- Sonic Boom
- numerous papers on applying and improving
- [Ku94] influential book on automata-theoretic ver'n
- [PR89] found'l paper on automata-theor'c synthesis

Amir Pnueli

* Seminal Ideas
- TL: right concept of concurrency
- TL: theor. sound, pract. useful framework

* Seismic Impact
- Tarski of Computer-Aided Verification