Al Aho

aho@cs.columbia.edu

The Quintessential Questions of Computer Science

40th Year Technical Symposium
Department of Computer Science
North Carolina State University
October 25, 2007

Warm-Up Question

- **What is the biggest impact that computer science has had on the world in the past forty years?**

- **My answer: the Internet and its associated global information infrastructure**

The 10 most popular programming languages in 1967

- **Algol 60**
- **APL**
- **Basic**
- **BCPL**
- **COBOL**
- **Fortran IV**
- **Lisp 1.5**
- **PL/I**
- **Simula 67**
- **SNOBOL 4**

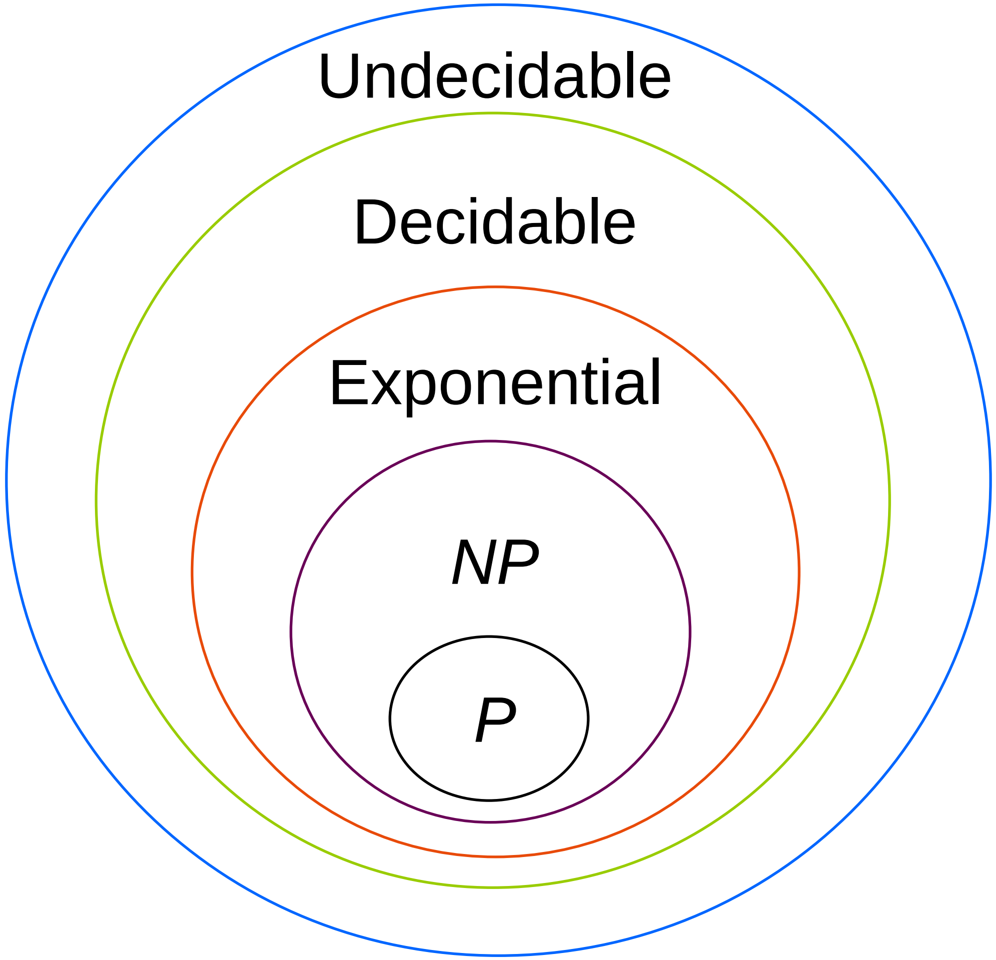
The 10 most popular programming languages in 2007

- Java
- C
- Visual Basic
- C++
- PHP
- Perl
- C#
- Python
- JavaScript
- Ruby

TIOBE PROGRAMMING COMMUNITY INDEX **October 2007**
www.tiobe.com

Question 1

- **How do we determine the difficulty of a problem?**



Complexity Hierarchy

The Classes *P* and *NP*

- **A problem is in *P* if it can be solved in polynomial time by a deterministic Turing machine.**

  Example: Does a set of *n* positive and negative integers have a nonempty subset whose sum is positive?

  { –2, 7, –3, 14, –10, 15 }

- **A problem is in *NP* if it can be solved in polynomial time by a nondeterministic Turing machine.**

  Example: Does a set of *n* positive and negative integers have a nonempty subset whose sum is zero?

  { –2, 7, –3, 14, –10, 15 }

The *P* vs. *NP* Problem

- **Does *P* = *NP*?**

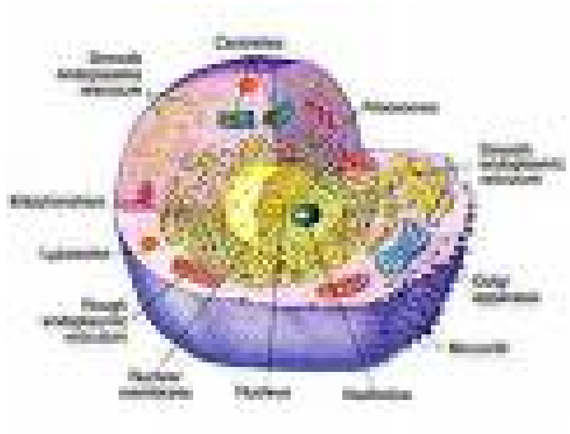
- **Informally: Are there any problems for which a computer can verify a given solution quickly but cannot find the solution quickly?**

- **Note: This is one of the Clay Mathematics Institute Millennium Prize Problems. The first person solving this problem will be awarded one million US dollars by the CMI (http://www.claymath.org/millennium).**

Question 2

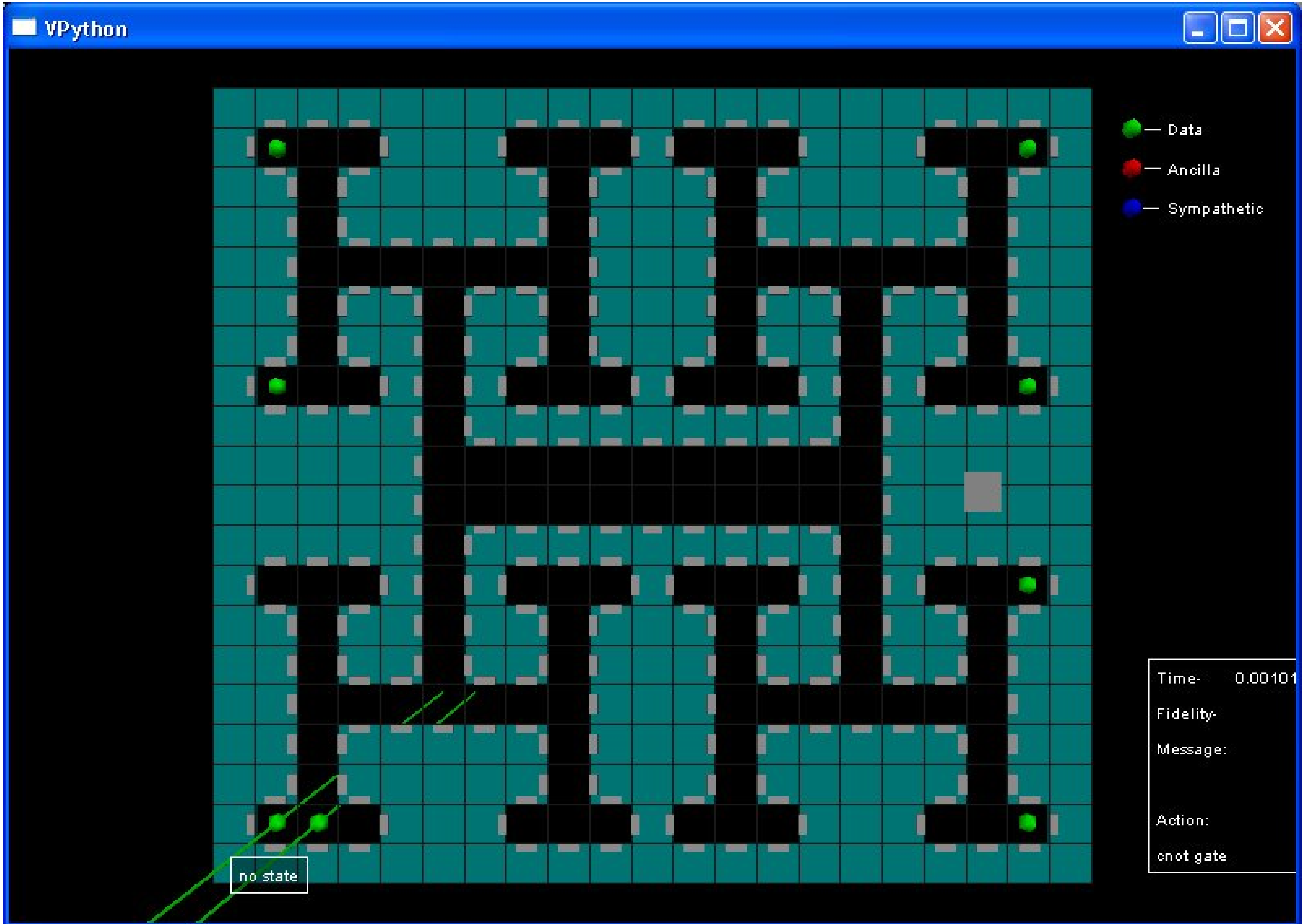
- **How do we model the behavior of complex systems that we would like to simulate?**

Large software systems

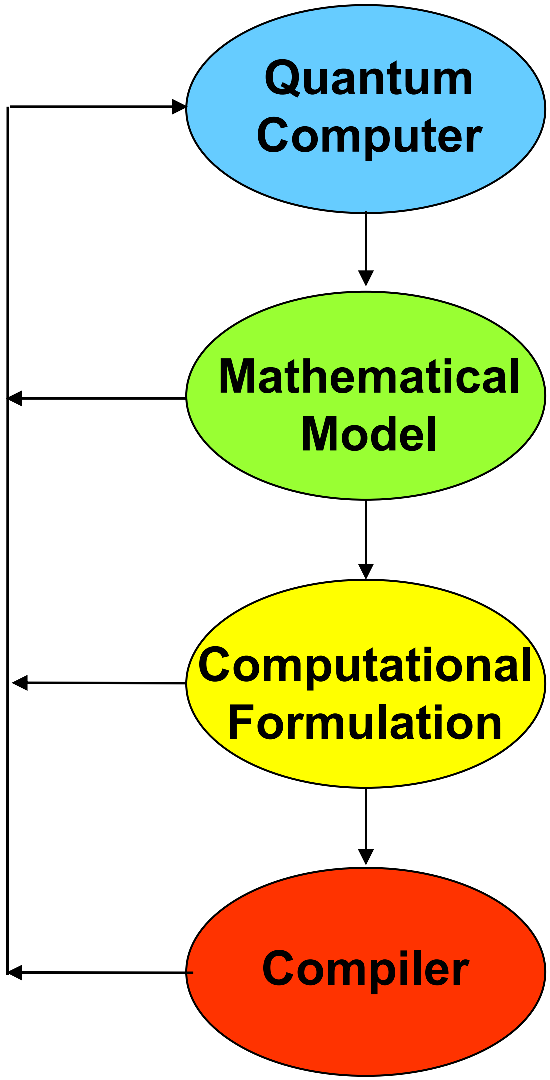


Human cell

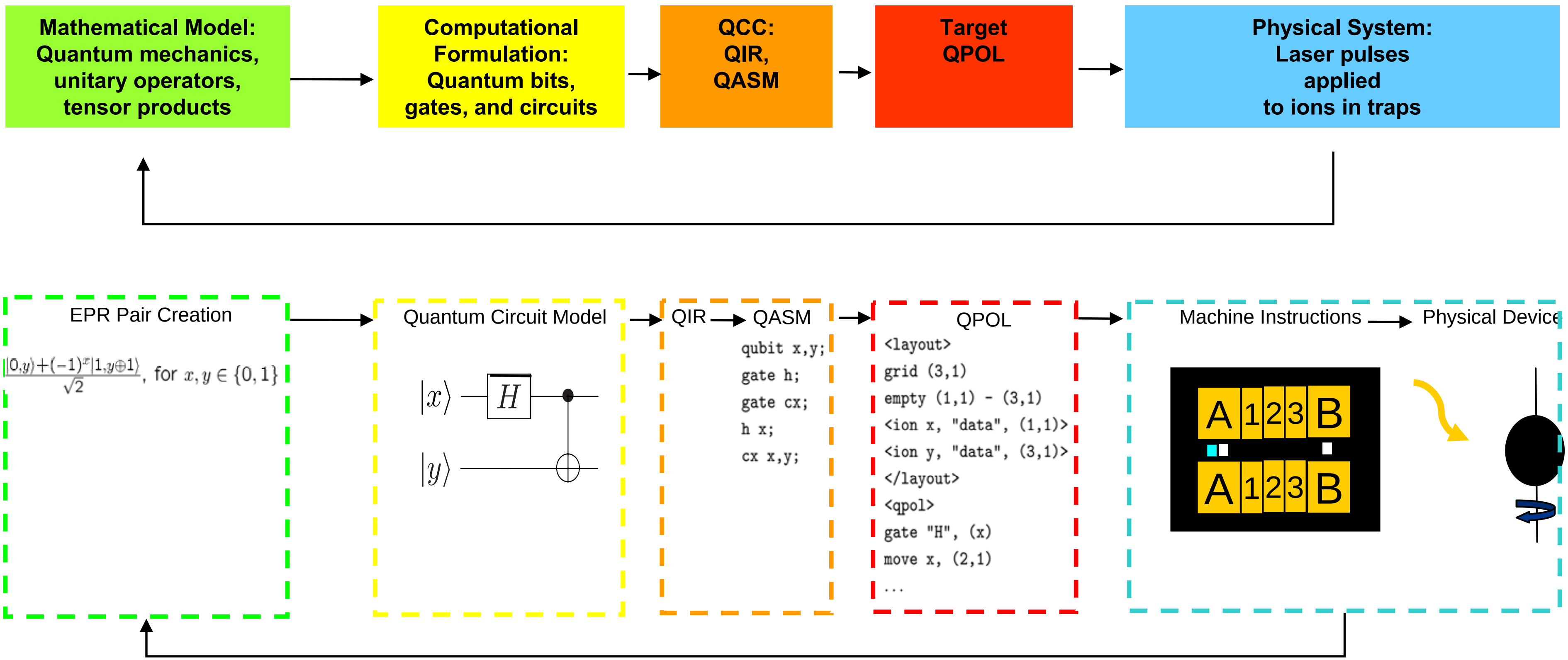
Ion Trap Quantum Computer



Programming Languages and Compilers for Quantum Computers

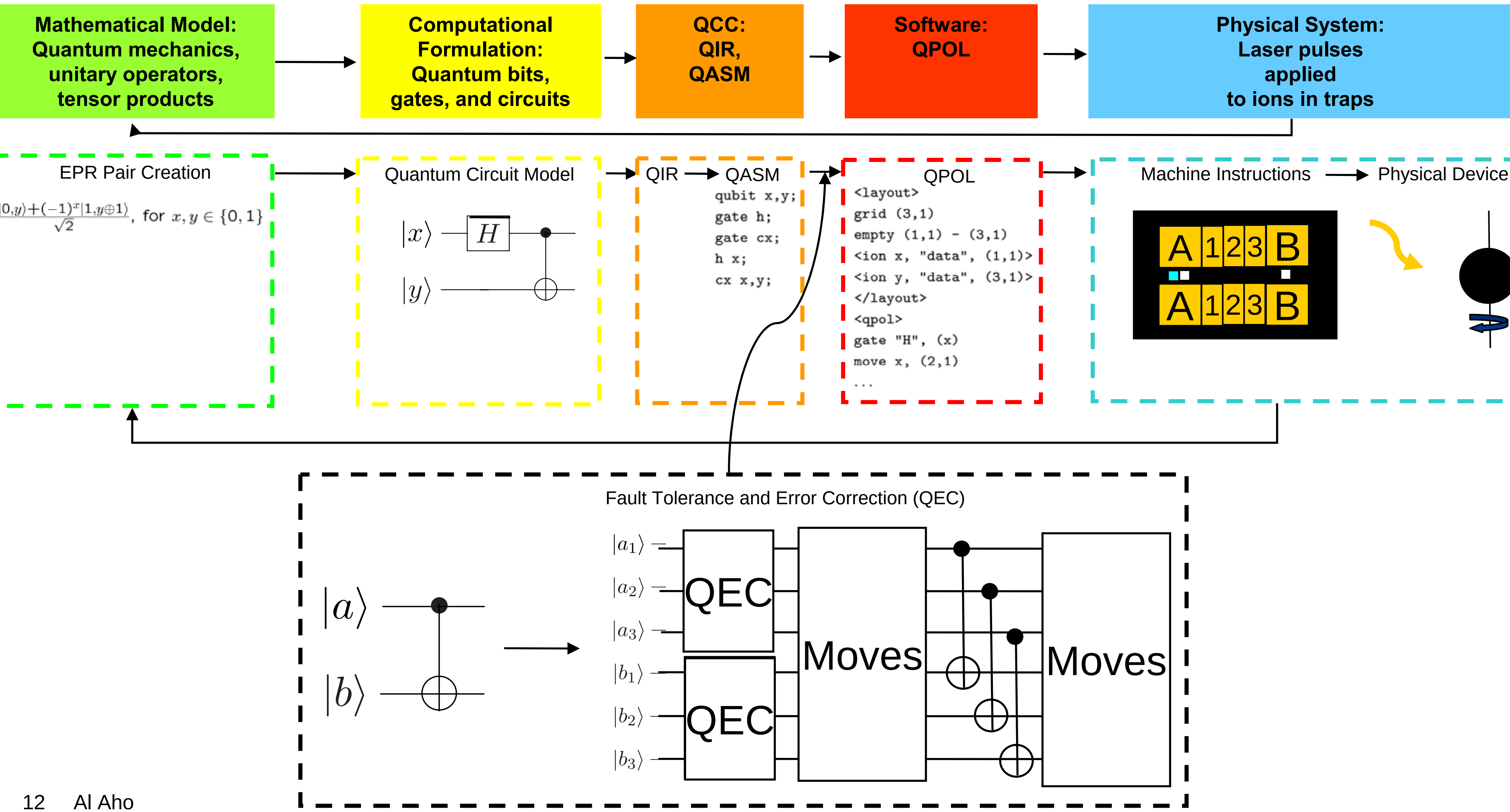


Quantum Computer Compiler



K. Svore, A. Aho, A. Cross, I. Chuang, I. Markov
A Layered Software Architecture for Quantum Computing Design Tools
IEEE Computer, 2006, vol. 39, no. 1, pp.74-83

Design Flow with Fault Tolerance and Error Correction



Question 3

- **How do we build a trustworthy information infrastructure?**

Demand for Trustworthy Systems

- 36 million Americans have had their identities stolen since 2003
- 155 million personal records have been compromised since 2005
- 28 million veterans had their Social Security numbers stolen from laptops

Annie I. Antón
Testimony before the Subcommittee on Social Security
U.S. House of Representatives Committee on Ways and Means
June 21, 2007

Demand for Trustworthy Systems
Protection from Malware

- **Internet malware**
  - **worms, viruses, spyware and Internet-cracking tools**
  - **worms override program control to execute malcode**
- **Internet worms**
  - **Morris '88, Code Red II '01, Nimda '01, Slapper '02, Blaster '03, MS-SQL Slammer '03, Sasser '04**
  - **automatic propagation**
- **Internet crackers**
  - “**j00** *got* **h4x0r3d**!!”
- **After breaking in, malware will**
  - **create backdoors, install root kits (conceal malcode existence), join a botnet, generate spam**

Worms, viruses prove costly

The estimated cleanup and lost productivity costs of worms and viruses add up:

Year	Virus/worm	Estimated damage
1999	Melissa virus	$80 million
2000	Love Bug virus	$10 billion
2001	Code Red I and II worms	$2.6 billion
2001	Nimda virus	$590 million to $2 billion
2002	Klez worm	$9 billion
2003	Slammer worm	$1 billion

Source: USA TODAY research

Gaurav S. Kc
Defending Software Against Process-Subversion Attacks
PhD Dissertation, Columbia University, 2005

Question 4

- **Is there a scientific basis for making reliable software?**

How Can We Make Reliable Software?

- **Communication:** Shannon [1948] used error detecting and correcting codes for reliable communication over noisy channels
- **Hardware:** von Neumann [1956] used redundancy to create reliable systems from unreliable components
- **Software:** Is there a scientific basis for making reliable software?

Volume of Software and Defects

- **World uses hundreds of billions of lines of software**
  - 5 million programmers worldwide
  - average programmer generates 5,000 new lines of code annually
  - embedded base: hundreds of billions of lines of software
- **Number of embedded defects**
  - defect densities: 10 to 10,000 defects/million lines of code
  - total number of defects in embedded base: 5×10^6 to 50×10^9

Alfred V. Aho, *Software and the Future of Programming Languages,* Science, February 27, 2004, pp. 1331-1333.

IEEE Spectrum Software Hall of Shame

Year	Company	Costs in US $
2004	UK Inland Revenue	Software errors contribute to $3.45 billion tax-credit overpayment
2004	J Sainsbury PLC [UK]	Supply chain management system abandoned after deployment costing $527M
2002	CIGNA Corp	Problems with CRM system contribute to $445M loss
1997	U. S. Internal Revenue Service	Tax modernization effort cancelled after $4 billion is spent
1994	U. S. Federal Aviation Administration	Advanced Automation System canceled after $2.6 billion is spent

R. N. Charette, *Why Software Fails,* IEEE Spectrum, September 2005.

The Software Development Process

- **Specification**
  - Define system functionality and constraints
- **Validation**
  - Ensure specification meets customer needs
  - "Are we building the right product?"
- **Development**
  - Produce software
- **Verification and testing**
  - Ensure the software does what the specification calls for
  - "Are we building the product right?"
- **Maintenance**
  - Evolve the software to meet changing customer needs
- **Quality plan**
  - Ensure product meets user needs

Where is the Time Spent?

- **1/3 planning**
- **1/6 coding**
- **1/4 component test and early system test**
- **1/4 system test, all components in hand**

"In examining conventionally scheduled projects, I have found that few allowed one-half of the projected schedule for testing, but that most did indeed spend half of the actual schedule for that purpose."

F. B. Brooks, *The Mythical Man-Month,* 1995.

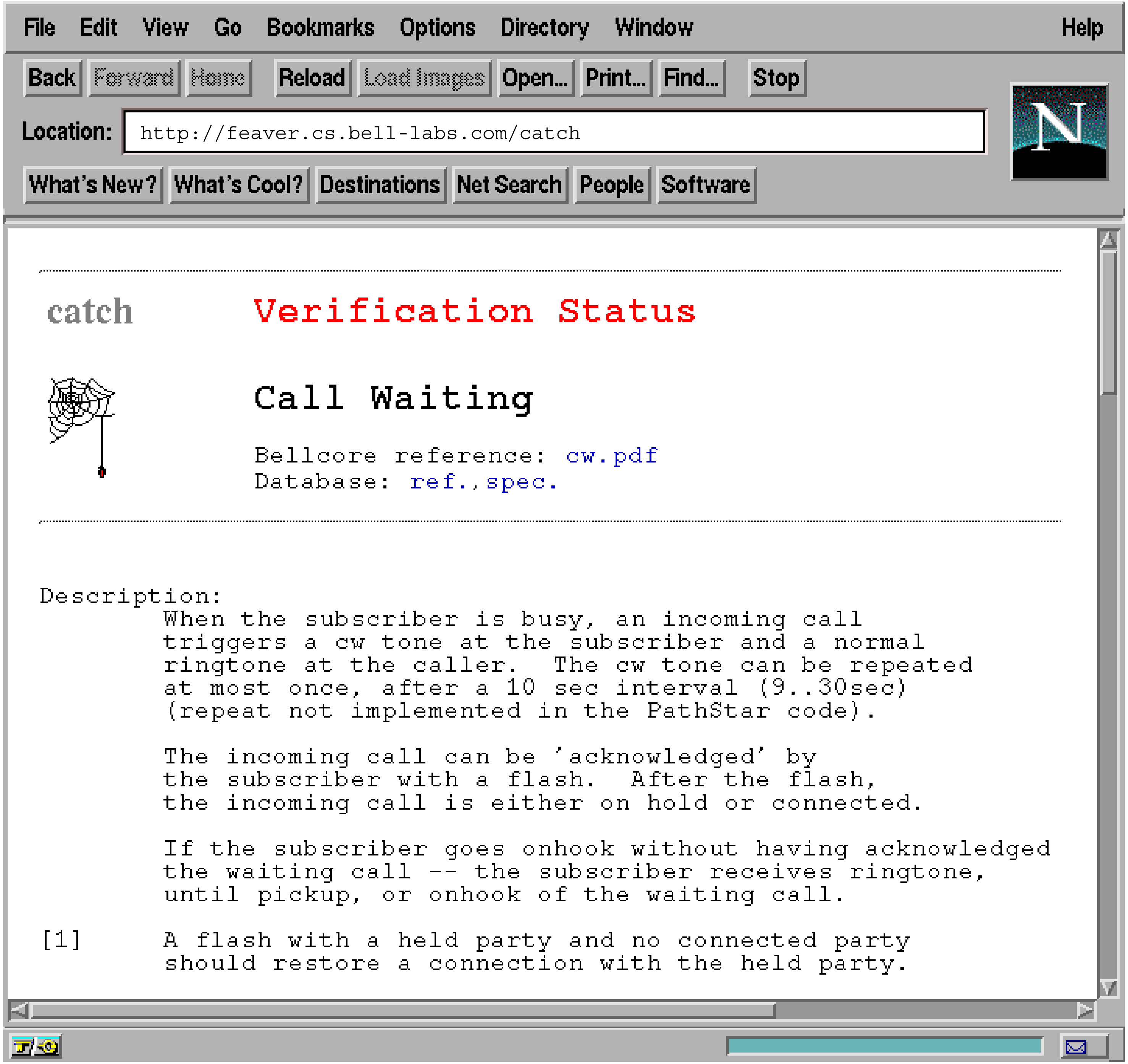
Why Do Software Projects Fail?

- Unrealistic or unarticulated project goals
- Inaccurate estimates of needed resources
- Badly defined system requirements
- Poor reporting of the project's status
- Unmanaged risks
- Poor communication among customers, developers, and users
- Use of immature technology
- Inability to handle the project's complexity
- Sloppy development practices
- Poor project management
- Stakeholder politics
- Commercial pressures

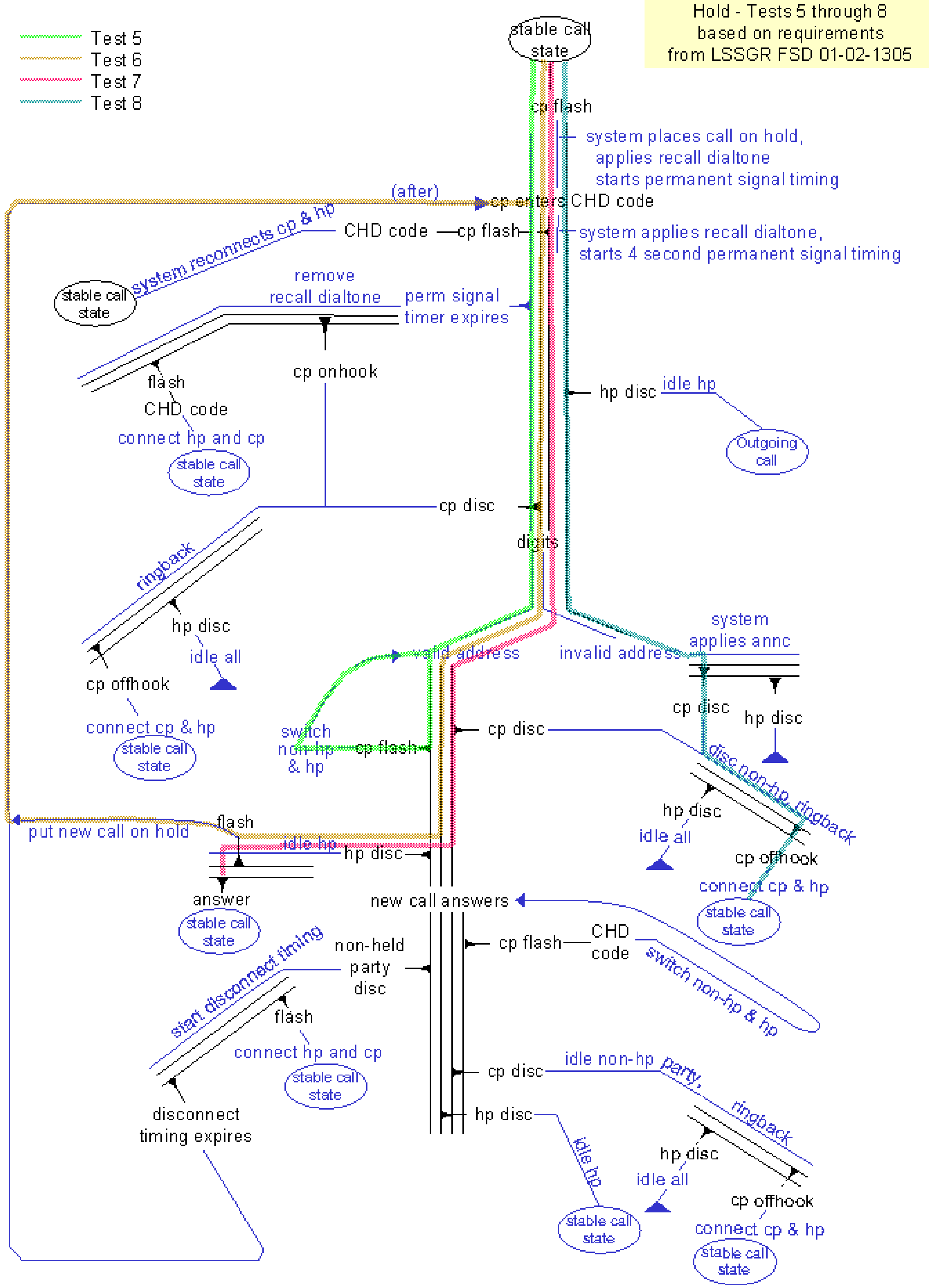
R. N. Charette, *Why Software Fails,* IEEE Spectrum, September 2005.

Ingredients for Making Reliable Software

- Good people/management/communication
- Good requirements/modeling/prototyping
- Sound software engineering practices
- Use of mature technology
- Thorough testing
- Verification tools
  - model checkers
  - theorem-proving static analyzers


File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Load Images Open... Print... Find... Stop
Location: http://feaver.cs.bell-labs.com/catch
What's New? What's Cool? Destinations Net Search People Software
catch
Verification Status
Call Waiting
Bellcore reference: cw.pdf
Database: ref., spec.
Description:
When the subscriber is busy, an incoming call triggers a cw tone at the subscriber and a normal ringtone at the caller. The cw tone can be repeated at most once, after a 10 sec interval (9..30sec) (repeat not implemented in the PathStar code).
The incoming call can be 'acknowledged' by the subscriber with a flash. After the flash, the incoming call is either on hold or connected.
If the subscriber goes onhook without having acknowledged the waiting call -- the subscriber receives ringtone, until pickup, or onhook of the waiting call.
[1] A flash with a held party and no connected party should restore a connection with the held party.

Modeling feature behavior



Every path through feature graph defines a system requirement and hence a check to be made.

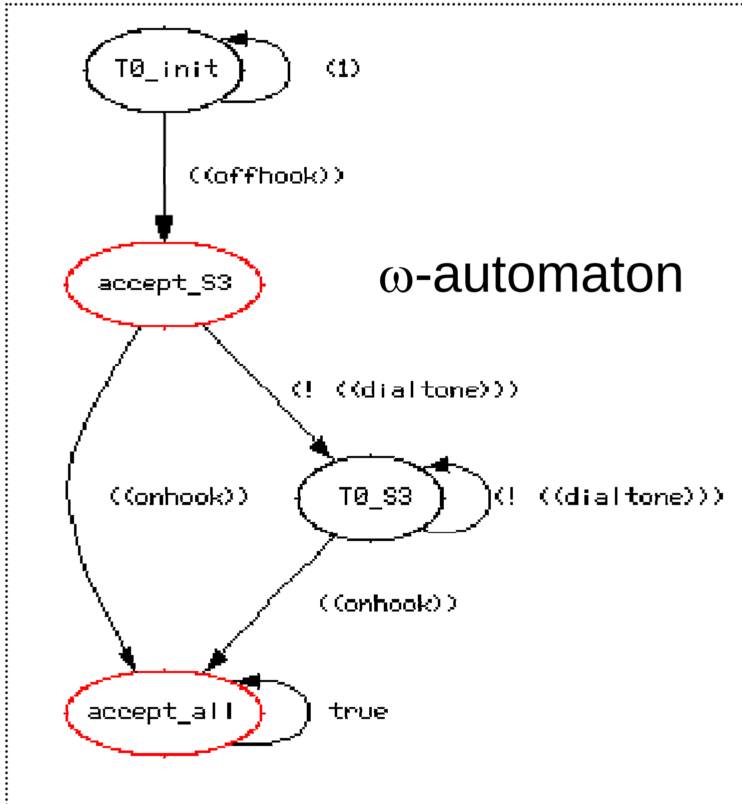
Modeling Requirements with Linear Temporal Logic

Example:

"When the subscriber goes offhook, dialtone is generated."

A failure to satisfy the property:

<>	**eventually,**
	the subscriber goes offhook
/	**and**
X	**thereafter, no dialtone is**
U	**generated until the next onhook**



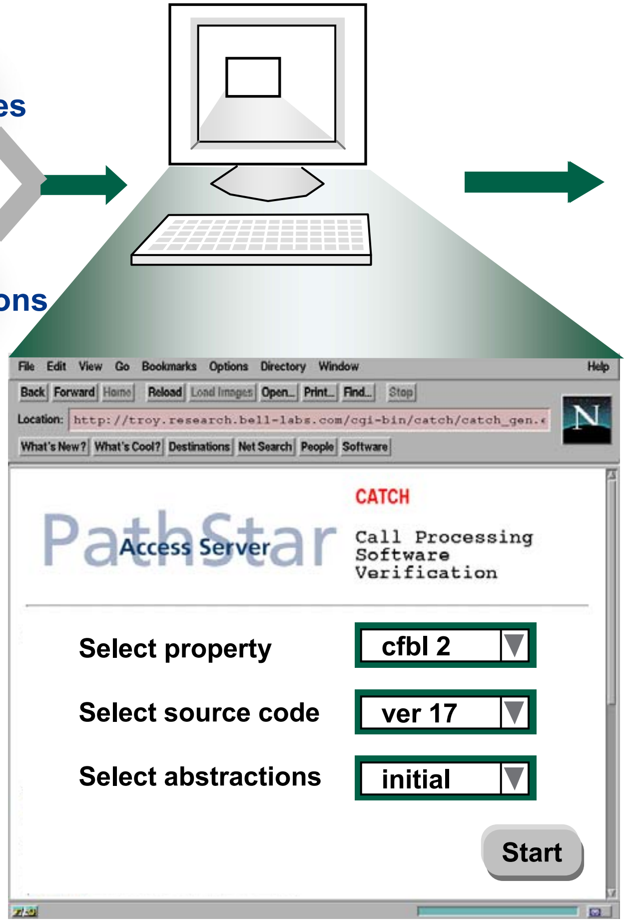
LTL formula:
<> (offhook /\\ X (!dialtone U onhook))

FeaVer Verification Process

Properties

Source code

Abstractions



Extract model from source code

Convert formulae to test automata

Compile executable model checker -- one per property

Schedule and execute checks using model checking engine *spin*

Generate error traces in source language

Gerard Holzmann
FeaVer Software Verification System

But the open problem remains

Is there a scientific basis for making reliable software?

Question 5

- **Can we construct computer systems that have human-like attributes such as emotion or intelligence?**

Cogito, ergo sum.

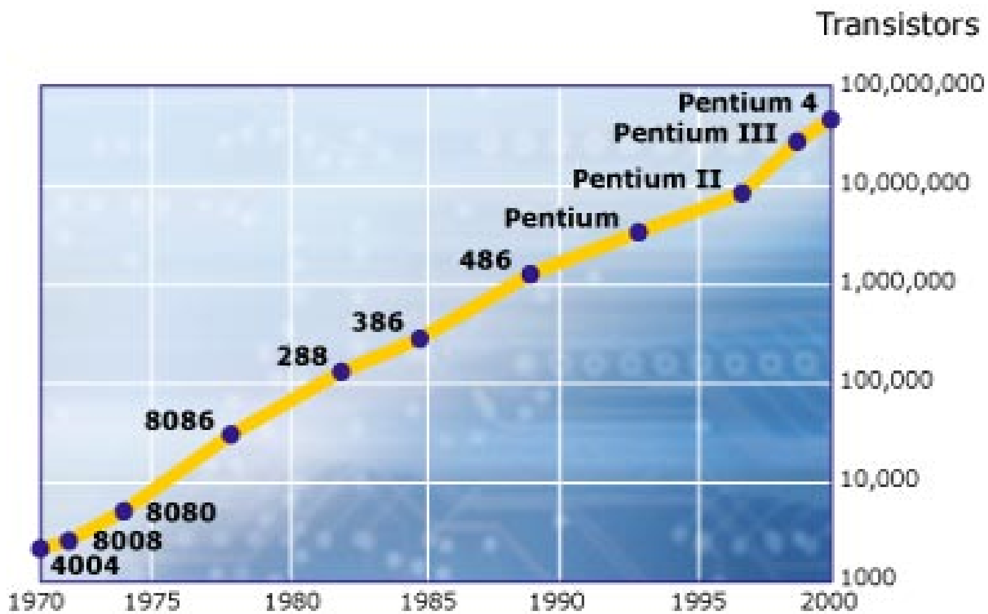
Marriage with Robots?

"My forecast is that around 2050, the state of Massachusetts will be the first jurisdiction to legalize marriages with robots."

David Levy
AI researcher
University of Maastricht, Netherlands
LiveScience, October 12, 2007

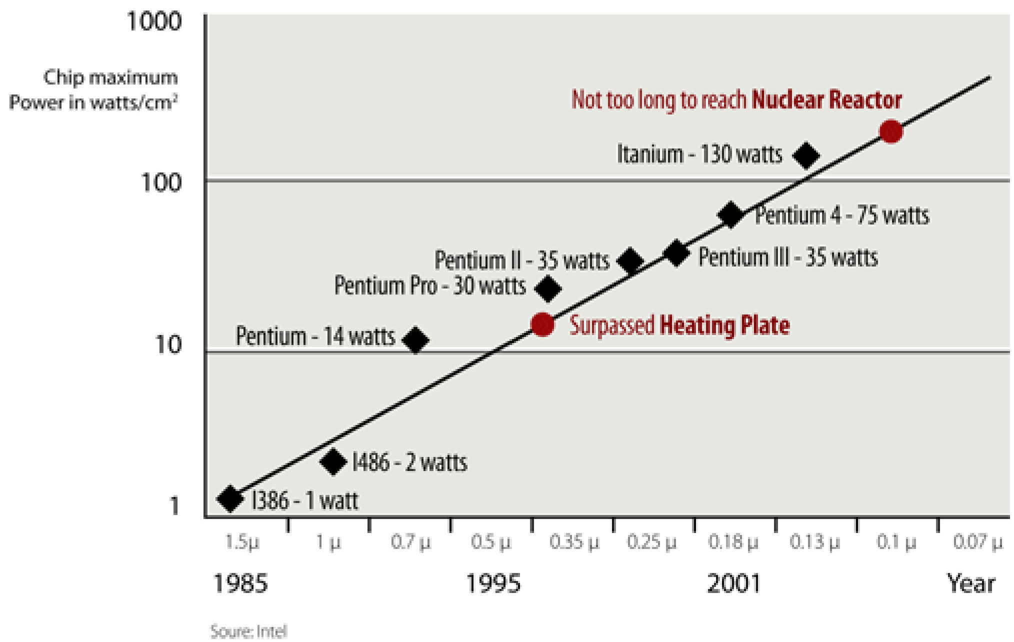
Bill Gates

Moore's Law for number of transistors on a chip



Bill Gates

Moore's Law for power consumption



Bill Gates's Question

- **How do we extend Moore's Law?**

- **Are multicore architectures the answer?**

Summary

1. **How do we determine the difficulty of a problem?**
2. **How do we model the behavior of complex systems that we would like to simulate?**
3. **How do we build a trustworthy information infrastructure?**
4. **Is there a scientific basis for making reliable software?**
5. **Can we construct computer systems that have human-like attributes such as emotion or intelligence?**
6. **How do we extend Moore's Law?**