Computer Science for the Smartphone Generation

Prof Aamod Sane
FLAME University, Pune

About the speaker

- Ph.D from U. of Illinois Urbana Champaign
- Architect for Search Backend, Semantic Search
- Designed many new systems, algorithms, some formal approaches for systems modelling
- Got interested in “how is mathematics created”
- Stepped back from industry to research the history of the discovery of mathematics, especially symbolic algebra and calculus 1580 to 1780
- Taught maths over 2 years to students ranging from 4^{th} grade, 8^{th} grade, JEE students and Masters students at various schools

An unusual opportunity

- A strange problem: starting a new CS department
- Not something I had done before, or known anyone similar
- Had been academically inclined, but not really a professional academic
- Advice from other academic / industrial colleagues
  - Anuradha Laxminaryanan, Jayaraman Valadi
- Experience with students ranging from 4th grade to Masters

Key considerations: The discipline and the students

- I had one strong impression from days in the industry
- Computer Science is becoming increasingly mathematical day by day
- And another, from teaching courses
- Students have grown up with the internet and especially smartphones

Student generations

- Desktop Generation – desktops usually not at home, only at school
- Laptop Generation – devices available at home
- Smartphone Generation – an extra organ
  - U.S. Supreme Court Chief Justice John Roberts
  - “such a pervasive and insistent part of daily life that the proverbial visitor from Mars might conclude they were an important feature of human anatomy.”

Student Generations Conception of Computing

- Desktop generation: Microsoft Apps and Games are Computing, occasional glimpses of programming,
- Laptop generation: Microsoft, plus many more sophisticated games, some access to programming
- Smartphone generation: Apps are wildly sophisticated, and crucially, Networked.

How do you startle / impress students of the Smartphone Generation?

- How do we not bore students?
- If our initial courses ask them to draw “stars on the screen”, should they not laugh at us? (Behind our backs??)
- Does it mean we must show them how to build apps in the first course?

My answer: Shoot for intellectual seriousness

- The first courses should convince students that there is far more to computing than programming languages and apps
- Get them much more interested in the behind-the-scenes part, and not be too impressed by the flash.

Evolution of the discipline

- One change in CS is obvious to us all
- Linear algebra and Statistics have become central
- AI techniques are to modern scientific practice what Calculus was to science in 19th and 20th centuries

A second change, also relatively well known

- The second change which is better understood is large scale data processing software
- Hadoop was the poster child a while ago, but this space is changing and getting packaged as services
- The crucial new ideas are the CAP theorem and related ideas

Another change is less obvious

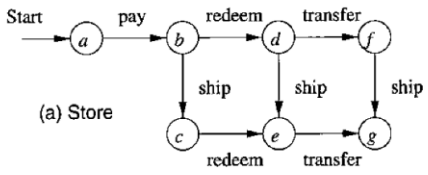
- Formal methods have come a long way
- High end programming jobs ask for skills in TLA and Alloy, and probably others as well.
- The underlying idea here is called Model Checking
- Lean is a tool that makes Logic into a programming language, in effect.
- It has proven itself in Maths and it or similar tools will become commonplace in a decade or so.

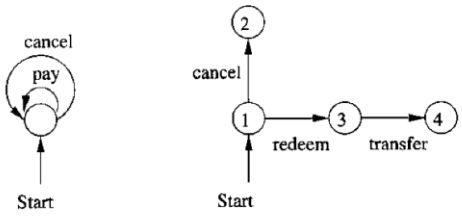
And another: Speculation about *Specification*

- As programming tools get more sophisticated, people will have to focus on PURPOSE of programming
- Specification will start dominating
- Algorithm design will be in the hands of people, implementation can be done by machines
- This is already happening
- Predictive programming tools from Github etc.
- ChatGPT will become an increasingly sophisticated programmer

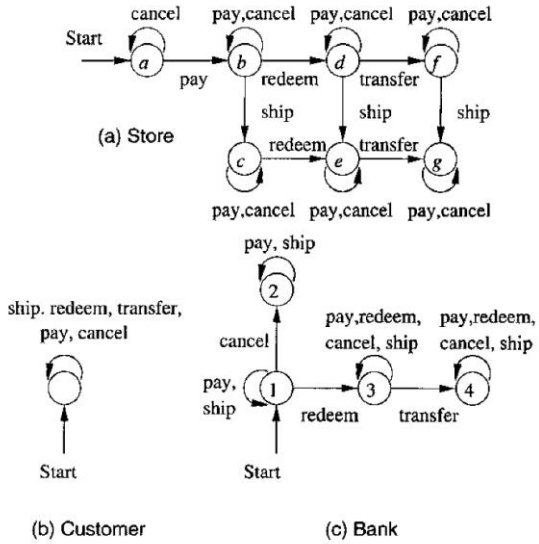
Prediction: Compiler Theory → Model Checking for Distributed Programs

- In the past, a key reason to learn formal tools like automata theory was its use in Parsing tools
- In the future, the key reason will be Model Checking
- This shift is already happening and will become more ingrained





(b) Customer (c) Bank



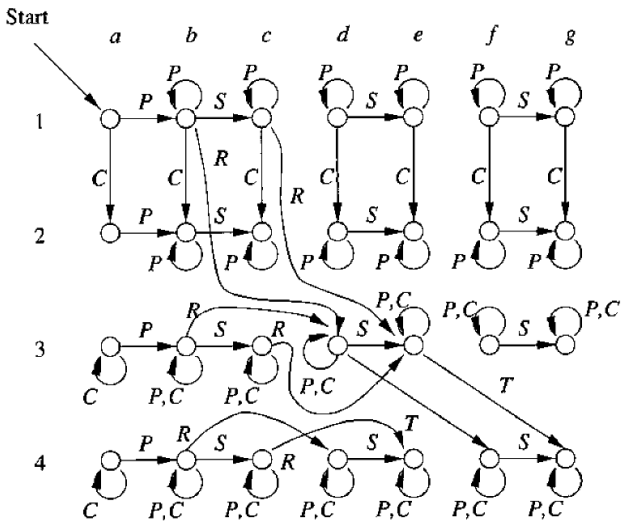


Figure 2.3: The product automaton for the store and bank

That was from…

- That was from Hopcroft, Ullman, Motwani (2001 !!)
- However, they do not pursue model checking in the text

Prediction: Specification techniques

- Lean is a tool for programming mathematics
- Instead of writing formulas on paper, you write them as programs
- The computer helps you prove them
- *Program properties are mathematical formulae*

What does Lean Look like?

```
def sum_of_first_n_nat : ℕ → ℕ | 0 := 0 | (nat.succ n) := (nat.succ n) + sum_of_first_n_nat n


theorem closed_eq_sum_of_first_n_nat (n : ℕ) : 2 * (sum_of_first_n_nat n) = n * (nat.succ n) :=
begin
end


case nat.zero ⊢ 2 * sum_of_first_n_nat 0 = 0 * 1
case nat.succ d : ℕ,
hd : 2 * sum_of_first_n_nat d = d * nat.succ d
⊢ 2 * sum_of_first_n_nat (nat.succ d) = nat.succ d * nat.succ (nat.succ d)
```

Still unfamiliar, but …

- Once we get used to it, and especially
- Our STUDENTS get used to it, being younger and more flexible they will force us to like it ☺

Our curriculum first year

- Key idea: Introduce significant ideas that will lead to AI and Formal Methods early, and build on them
- Introduction to Programming by Specification
  - This course uses Gofer, a functional language, in which at least for simple programs, there is little difference between a program and a specification
- Introduction to Computational Modeling
  - First half: Show how to model phenomena like Compound Interest, Projectile Motion etc. using programs
  - Second half: Elementary Neural Networks, simple image classification
- Introduction to Internet of Things
  - Use Arduino kits to build small artifacts

"Shoot for intellectual seriousness"

- The first courses should convince students that there is far more to computing than programming languages and apps
- Get them much more interested in the behind-the-scenes part, and not be too impressed by the flash.
- This part seems to work. Most students have never seen anything like Gofer, and find Computational Modeling as well as Arduino's captivating.

Curriculum by dependencies: Formal Methods Track

- Second year:
  - Sem 3 Mathematics for Computer Science (same as book)
  - Sem 4 Theory of Computation
- Third year:
  - Sem 5 Applied Formal Methods
- This sequence provides background for
  - Sem 7 : Fundamentals of Distributed Systems
  - Sem 8 : Cloud Infrastructure and Devops

Curriculum by dependencies: AI Track

- Second year:
  - Sem 3 Linear Algebra for Computer Science
  - Sem 4 Introduction to Probability and Statistics
- Third year:
  - Sem 5 Principles of Machine Learning
- Fourth year
  - Sem 7 : Advanced Machine Learning

Curriculum by dependencies: Systems track

- Second year:
  - Sem 3 C++
  - Sem 3 Computer Organization
  - Sem 4 Fundamentals of Computer Architecture
  - Sem 4 Systems Programming
- Third year:
  - Sem 5 Operating Systems
- This sequence provides background for
  - Sem 6 : Graphics
  - Sem 6 : Databases
  - Sem 6 : Web Systems

An unusual course? Systems Programming

- This course looks at details of how operating systems and applications interact. We study compilers, linkers, loaders, how system calls are organized, low level memory management. This course also looks at low level data structure implementations in machine level languages like C. We study the storage of data on file systems including storage structures used in database systems.

Another unusual choice? Web systems

- Instead of a standalone networking course, we blend basic ideas of networking with Web Systems
- The “network stack” TCP / DNS is introduced in the context of web systems design
- Browser architecture is also discussed.

Other standard courses

- Algorithms:
  - After C++ and Discrete Mathis in Sem 3,
  - Sem 4 Data Structures and Sem 5 Algorithms
- Software Architecture and Engineering
  - One course in Sem 7
- Security in Sem 8
- Electives in Sem 7 and 8: Compilers and Languages, Complexity Theory

Another one? Applied Formal Methods

- This course explores principles and practices of modern formal methods tools such as SAT Solvers and Model Checkers,

Constraints and Regrets

- Major plus Minor combination, 4 courses max per sem
- What to leave out?
  - Compilers and Languages is an Elective
    - No room to talk about Prolog, APL etc. ☹
  - Could not find room for Advanced Algorithms
  - A Mathematical Optimization course is an Elective, it could have dovetailed into the ML sequence
  - No standalone networks course:

Experience so far

- Our first batch is just in its final year
- Unfortunately, the first batch and Covid started in the same year, so there have been some ups and downs
- The subsequent batches seem to be going well; we will report real data at some point.