Theories of Programming Languages – Introduction

Xinyu Feng (冯新宇)

USTC

Acknowledgments: Many slides are taken from Zhong Shao's slides for Yale Formal Semantics class.

What is the course about?

- Similar features in different languages
  - Functions, variables, exceptions, types …
  - Why are they useful? What are the meanings?
  - What are good features, and what are bad ones?
    - “goto is harmful”, why?

- Many different languages
  - Imperative languages, OO languages, functional languages
    - What are the key features of each language?
    - When to use which?

We’ll try to answer question like:

- How to describe meanings of programs?
- How to describe properties of programs?
- How to reason about programs?
- How to tell if two programs have the same behaviors or not?
- How to design a new language?

Why take this course?

- *Software reliability and security are the biggest problems faced by the IT industry today!*

You are likely to worry about them in your future job!

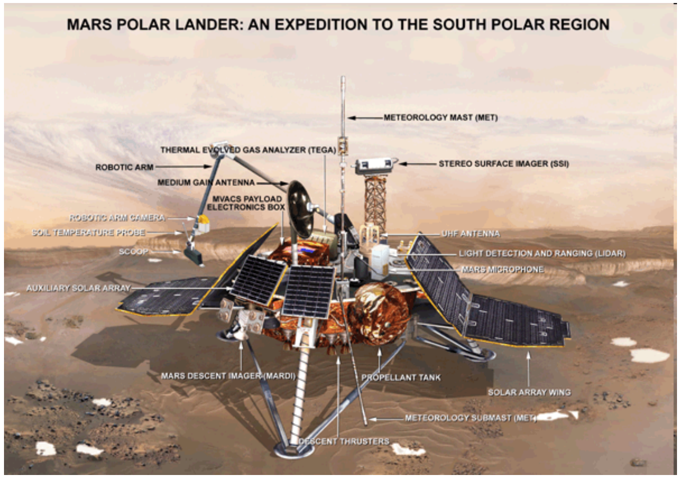
Arianne 5

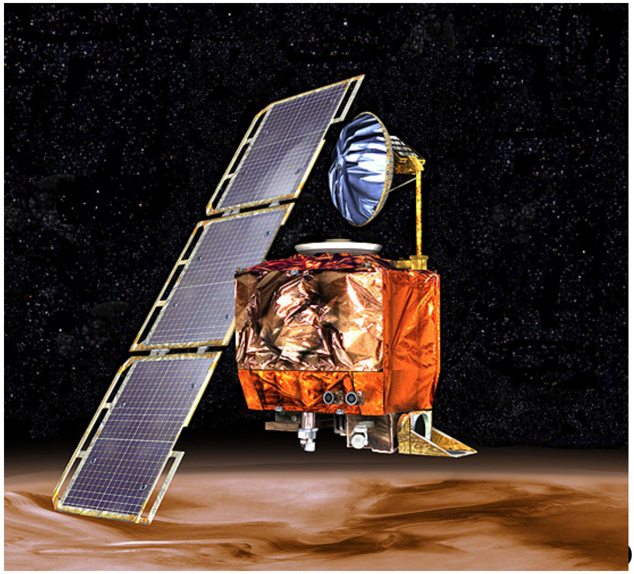
- On June 4, 1996, the Arianne 5 took off on its maiden flight.

- 40 seconds into its flight it veered off course and exploded.
- "Conversion of a 64bit integer into a 16bit signed integer leads to an overflow."

- This picture become quite popular in talks on software reliability

“Better, Faster, Cheaper”


MARS POLAR LANDER: AN EXPEDITION TO THE SOUTH POLAR REGION
METEOROLOGY MAST (MET)
THERMAL EVOLVED GAS ANALYZER (TEGA)
STEREO SURFACE IMAGER (SSI)
ROBOTIC ARM
MEDIUM GAIN ANTENNA
MVACS PAYLOAD ELECTRONICS BOX
ROBOTIC ARM CAMERA
SOIL TEMPERATURE PROBE
SCOOP
UHF ANTENNA
LIGHT DETECTION AND RANGING (LIDAR)
MARS MICROPHONE
AUXILIARY SOLAR ARRAY
MARS DESCENT IMAGER (MARDI)
PROPELLANT TANK
SOLAR ARRAY WING
METEOROLOGY SUBMAST (MET)
DESCENT THRUSTERS



- In 1999, NASA lost both the Mars Polar Lander and the Climate Orbiter.
- Later investigations determined software errors were to blame.
  - Orbiter: Component reuse error.
  - Lander: Precondition violation.

Therac-25

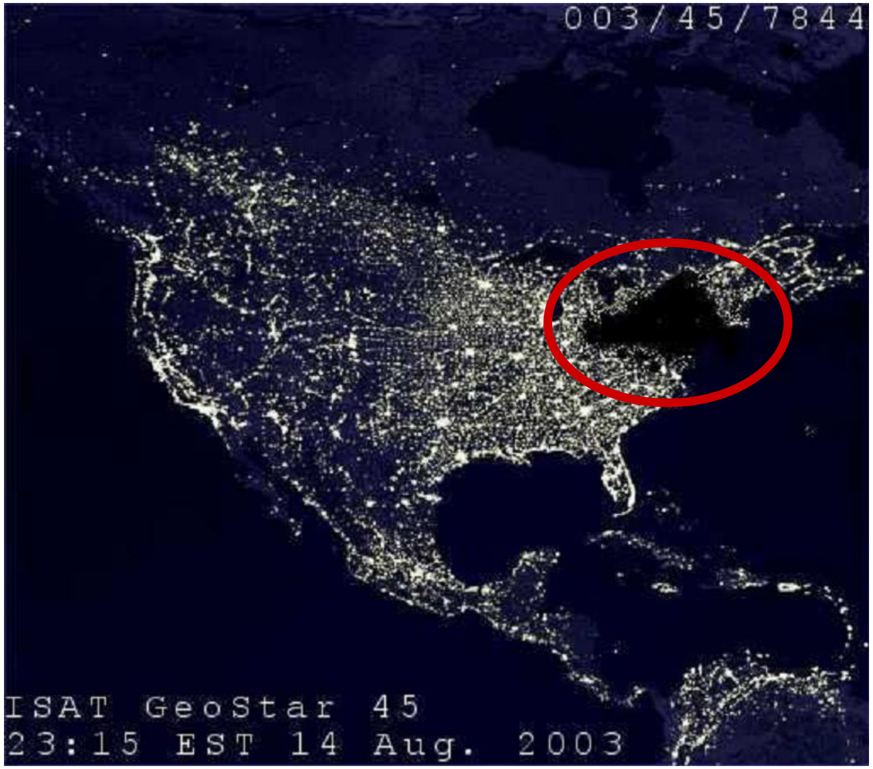
From 1985-1987, 6 patients were killed or seriously injured as a result of overdosed radiation (100 times of the intended dose) by Therac-25, a radiation treatment facility.

The problem was due to a subtle race condition between concurrent processes.

Northeast blackout, 2003

A widespread power outage that occurred throughout parts of the Northeastern and Midwestern United States and Ontario, Canada on Thursday, August 14, 2003

Race conditions in GE Energy's Unix-based XA/21 energy management system caused alarm system failure.



Crash of Wenzhou High-speed train

- Bugs in software controlling the signal system?

Now think of viruses and Trojan Horses

Stuxnet is used to attack the nuclear power station in Iran in 2010.

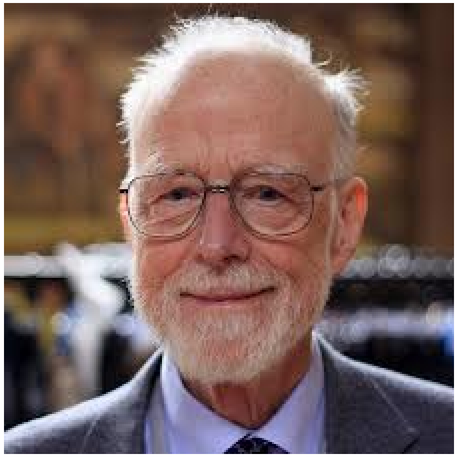
The virus took advantage of 4 undeclared bugs in windows to take over the system.

Bug-Free Software?

- A grand challenge for computer scientists
  - Posed since 1960’s
  - Significant progress, but still challenging

- Great practical implication
  - Software bugs cause the loss of 59.6 billion US dollars each year (0.6% GDP)
    - 2002 report from NIST
  - “Null References: The Billion Dollar Mistake”

    - Tony Hoare



Observations

- Failure often due to simple problems “in the details”.

- Small theorems about large programs would be useful.

- Need clearly specified interfaces and checking of interface compliance.

- Better languages would help!

New Challenges

Software is becoming more complex nowadays:

- Multi-core software
  - Concurrency
- Embedded software
  - Limited resources
- Distributed and cloud computing
  - Network environment
- Ubiquitous computing and Internet of Things

Opportunities

High assurance / reliability depends fundamentally on our ability to reason about programs.

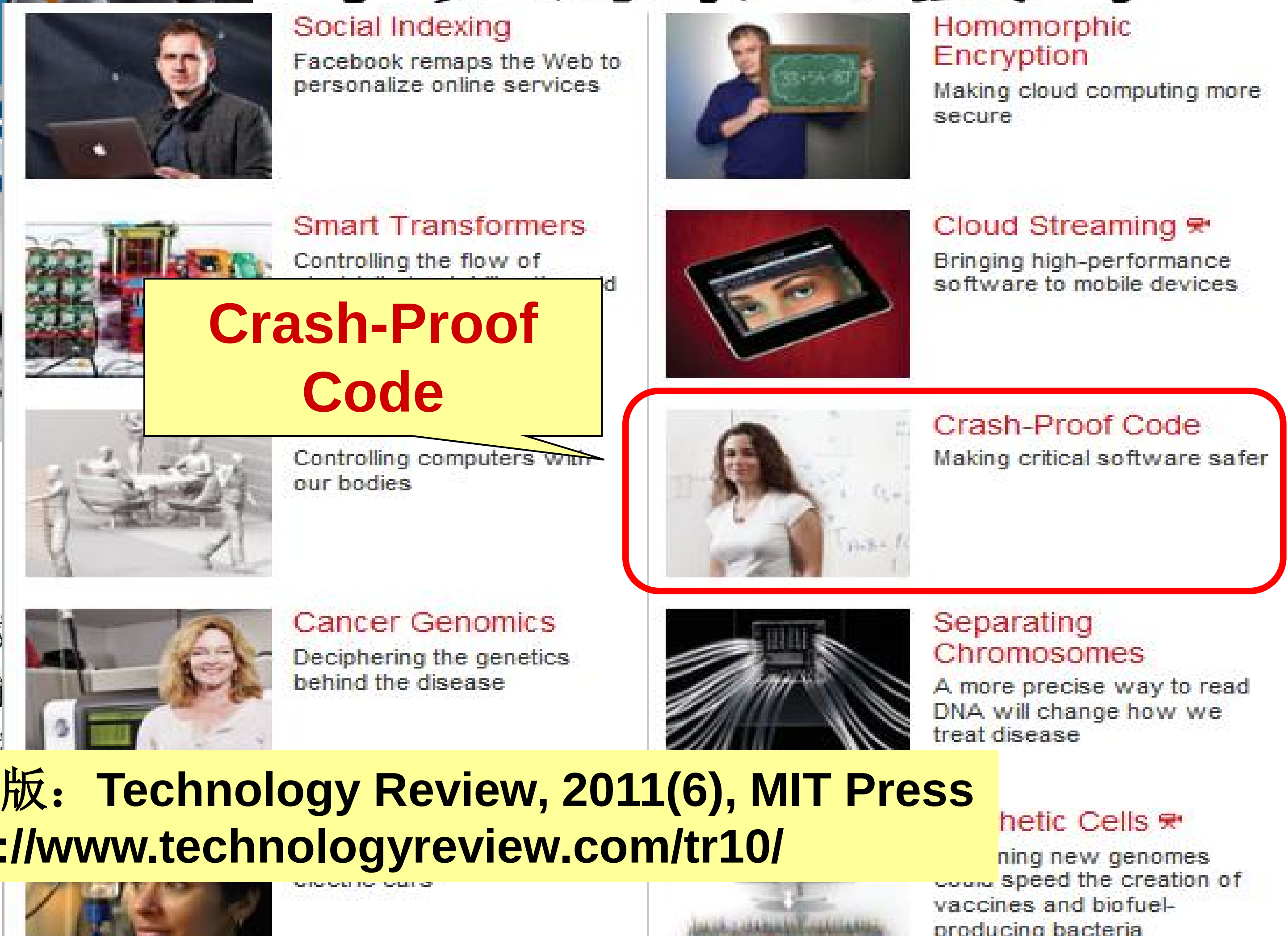
The opportunities for new languages as well as formal semantics, type theory, computational logic, and so on, are great.

2011世界
十大新兴技术

技术将会改变你的行为方式：你将用身体姿势来操控电视、
术可以促进你的健康，例如医生们将对不同肿瘤的相关基
出更有效的癌症疗法。不管技术属于哪一个类别，它们的
更加美好。

防崩溃代码

2011世界十大新兴技术


Social Indexing
Facebook remaps the Web to personalize online services

Homomorphic Encryption
Making cloud computing more secure

Smart Transformers
Controlling the flow of

Cloud Streaming
Bringing high-performance software to mobile devices

Controlling computers with our bodies

Crash-Proof Code
Making critical software safer

Cancer Genomics
Deciphering the genetics behind the disease

Separating Chromosomes
A more precise way to read DNA will change how we treat disease

electric cars

hetic Cells
ning new genomes could speed the creation of vaccines and biofuel-producing bacteria

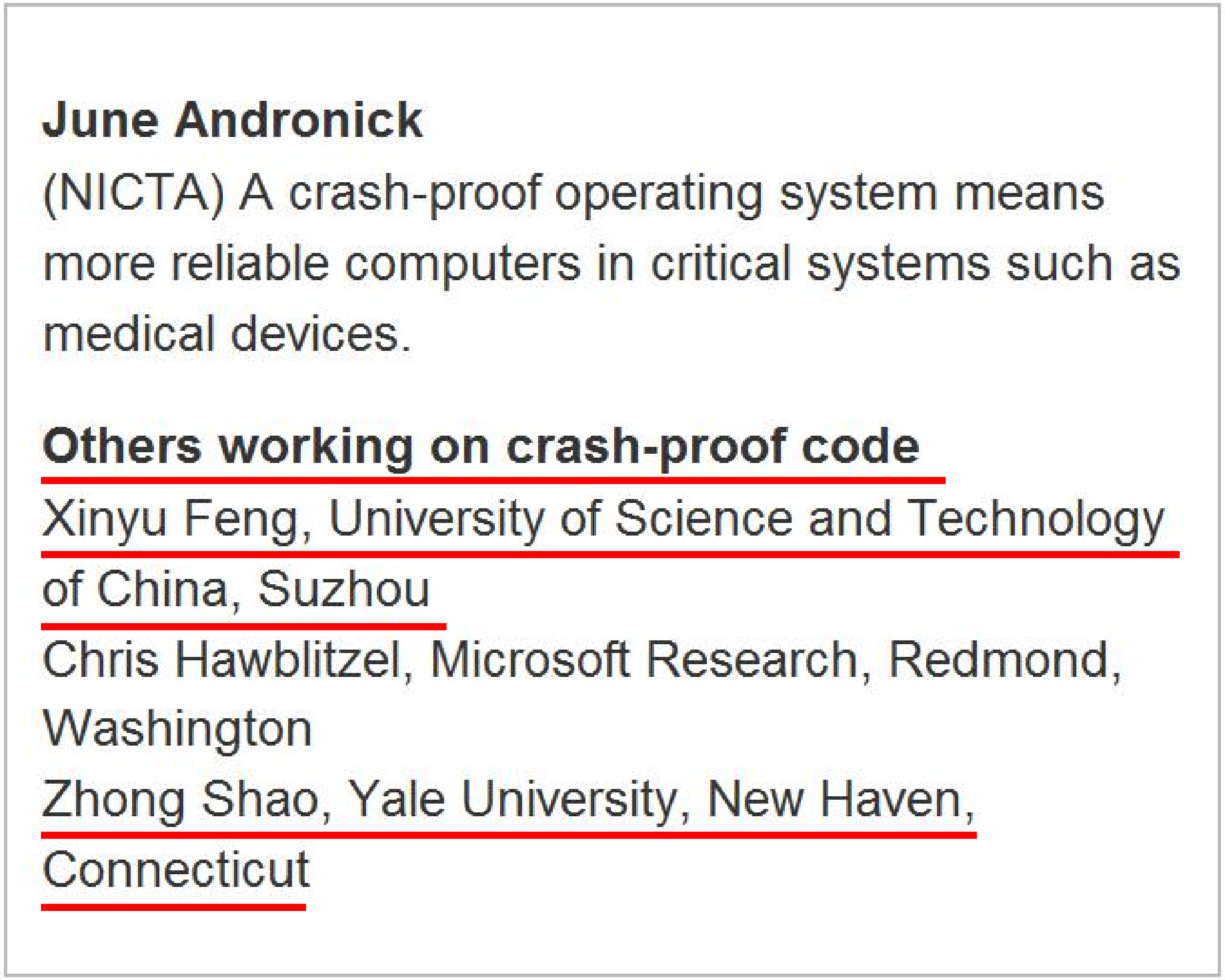
40 社交
42 智
44 手

Crash-Proof Code

英文版：Technology Review, 2011(6), MIT Press
http://www.technologyreview.com/tr10/

Report on Crash-Proof Code

- Verification of the seL4 OS kernel
  - Done by the Australian group at NICTA
  - Give mathematical proof showing the kernel would never crash
    - How to do this mathematically?
    - How to define “crash”?
    - How to prove the system is “crash-proof”?
    - We will answer the questions in this course



June Andronick

(NICTA) A crash-proof operating system means more reliable computers in critical systems such as medical devices.

Others working on crash-proof code

Xinyu Feng, University of Science and Technology of China, Suzhou

Chris Hawblitzel, Microsoft Research, Redmond, Washington

Zhong Shao, Yale University, New Haven, Connecticut

界

兴技术

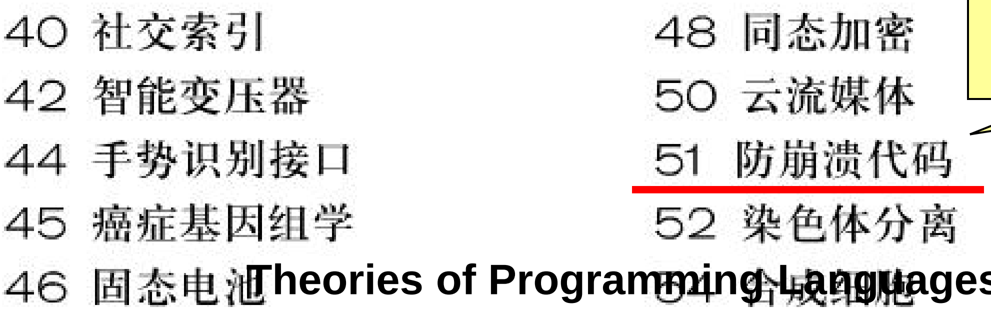
将用身体姿势来操控电视、

生们将对不同肿瘤的相关基

术属于哪一个类别，它们的

防崩溃代码

(Crash-Proof Code)



40 社交索引

42 智能变压器

44 手势识别接口

45 癌症基因组学

46 固态电池

48 同态加密

50 云流媒体

51 防崩溃代码

52 染色体分离

54 合成细胞

Why take this course (2)

- Software reliability and security are the biggest problems faced by the IT industry today! You are likely to worry about them in your future job!
- It will give you an edge over your competitors: **industry and most other schools don’t teach this**.
- It will improve your programming skills – because you will have a better appreciation of what your programs actually *mean*.
- You will be better able to compare and contrast programming languages, or even design your own.
- It’s intellectually deep: there’re many challenging and hot research problems.

Course Overview

Goals of the Course

- Survey existing language features
  - What they mean? What they do? How they compare?
- Methods to define behaviors of programs
  - Operational/Denotational/Axiomatic Semantics
- Methods to reason about properties of programs
  - Define and prove “correctness” of programs
  - Building “crash-proof” or “bug-free” software

Preliminary Syllabus

- Introduction
- Functional programming and Coq
- Predicate logic
- Imp and its semantics
  - Denotational, operational, and axiomatic
- Lambda calculus
- Type systems
- C-like pointer programs and separation logic

Preliminary Syllabus (2)

- Concurrency
- Process calculus
- Advanced topics
  - Program equivalence
  - Linearizability
  - OS verification
  - Compiler verification

Course Requirements

- Class attendance is highly recommended
- Homework
  - Problem sets & Paper reviews
  - Programming assignments in Coq
- Readings
  - Lecture notes and textbooks
  - Some research papers
  - Tutorials on Coq
- ***Grading***
  - 40% attendance and homework (will be helpful if you make me know you)
  - 60% final exam or projects (TBD, no mid-term exam)

Course Requirements (2)

It will be easy to everyone to pass the exam, but

Absolutely NO cheating and plagiarism!

Course Webpage

http://staff.ustc.edu.cn/~xyfeng/teaching/TOPL/

Notifications, text books and tools, reading materials, lecture notes and homework will all be posted on the webpage.

Please pay attention to the updates.