دانشگاه آزاد اسلامی واحد تبریز

نام درس: روش های رسمی در مهندسی نرم افزار

بخش: مقدمه ای بر روش های رسمی در مهندسی نرم افزار

نام استاد: دکتر مسعود کارگر

هدف از درس

1. Learn about formal methods (FM) in software engineering
2. Understand how formal methods (FM) help produce high-quality software
3. Learn about formal modeling and specification languages
4. Write and understand formal requirement specifications
5. Learn about main approaches in formal software verification
6. Know which formal methods to use and when
7. Use automatic and interactive tools to verify models and code

موضوعات درس

Software Specification

- High-level design
- System-level design (Model-based Development)
- Code-level design

Main Software Validation Techniques

- Model Finding/Checking:
  often automatic, abstract
- Deductive Verification:
  typically semi-automatic, precise (source code level)
- Abstract Interpretation:
  automatic, correct, incomplete, terminating

سازماندهی درس

- Course organized by level of specification
- Emphasis on tool-based specification and validation methods
- A number of ungraded exercises
- Hands-on homework where you specify, design, and verify
- For each main topic
  - A team introductory homework assignment
  - A team mini-project
- More details on the syllabus and the website

قسمت ۱: طراحی سطح بالا

Language: Alloy

- Lightweight modeling language for software design
- Amenable to a fully automatic analysis
- Aimed at expressing complex structural constraints and behavior in a software system
- Intuitive structural modeling tool based on first-order logic
- Automatic analyzer based on SAT solving technology

Learning Outcomes

- Design and model software systems in the Alloy language
- Check models and their properties with the Alloy Analyzer
- Understand what can and cannot be expressed in Alloy

قسمت دوم: توسعه مبتنی بر مدل

Language: Lustre

- Executable specification language for synchronous reactive systems
- Designed for efficient compilation and formal verification
- Used in safety-critical applications industry
- Automatic analysis with tools based on model-checking techniques

Learning Outcomes:

- Write system and property specifications in Lustre
- Perform simulations and verifications of Lustre models
- Understand what can and cannot be expressed in Lustre

قسمت ۳: مشخصات سطح کد

Language: Dafny

- Programming language with specification constructs
- Specifications embedded in source code as formal contracts
- Tool support with sophisticated verification engines
- Automatic analysis based on theorem proving techniques

Learning Outcomes:

- Write formal specifications and contracts in Dafny
- Verify functional properties of Dafny programs with automated tools
- Understand what can and cannot be expressed in Dafny

قسمت چهارم (اضافی): اثبات قضیه تعاملی

Language: Lean

- General-purpose logical language with executable sublanguage
- Supports both mathematical reasoning and reasoning about complex systems
- Powerful proof-assistant
- Semi-automatic, based on interactive theorem proving techniques

Learning Outcomes:

- Write formal specifications and programs in Lean
- Proof properties in Lean interactively
- Understand what can and cannot be expressed in Lean

یک حقیقت

Software has become critical to modern life

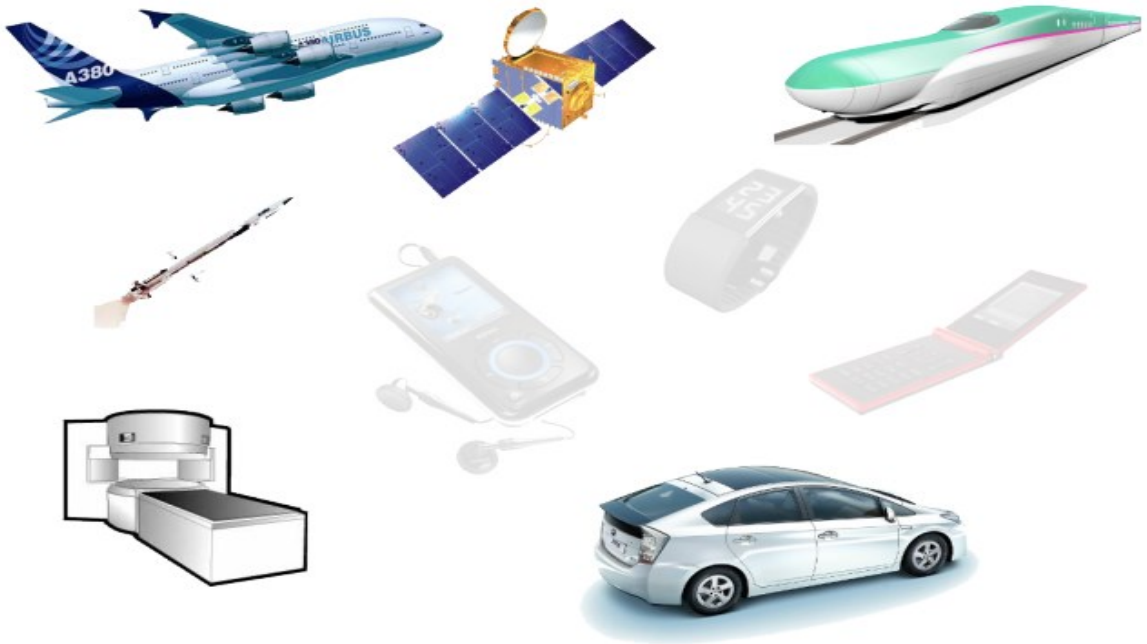
- Communication (internet, voice, video, . . .)
- Transportation (air traffic control, avionics, cars, . . .)
- Health Care (patient monitoring, device control, . . .)
- Finance (automatic trading, banking, . . .)
- Defense (intelligence, weapons control, . . .)
- Manufacturing (precision milling, assembly, . . .)
- Process Control (oil, gas, water, . . .)
- . . .

نرم‌افزار تعبیه‌شده

Software is now embedded everywhere

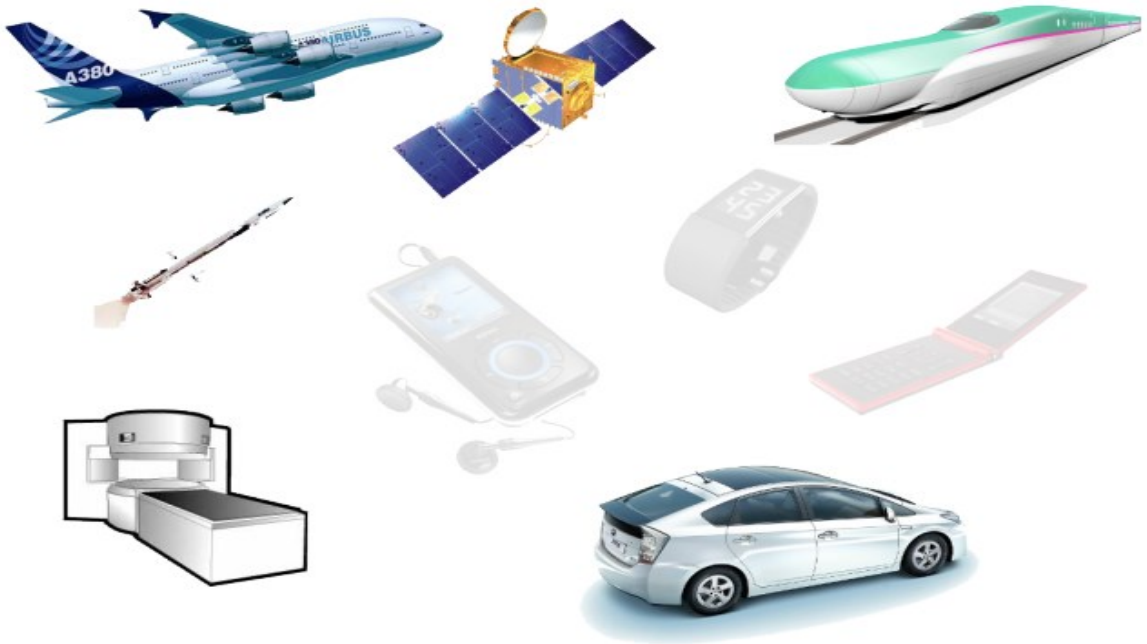
نرم‌افزار تعبیه‌شده

Software is now embedded everywhere Some of it is critical



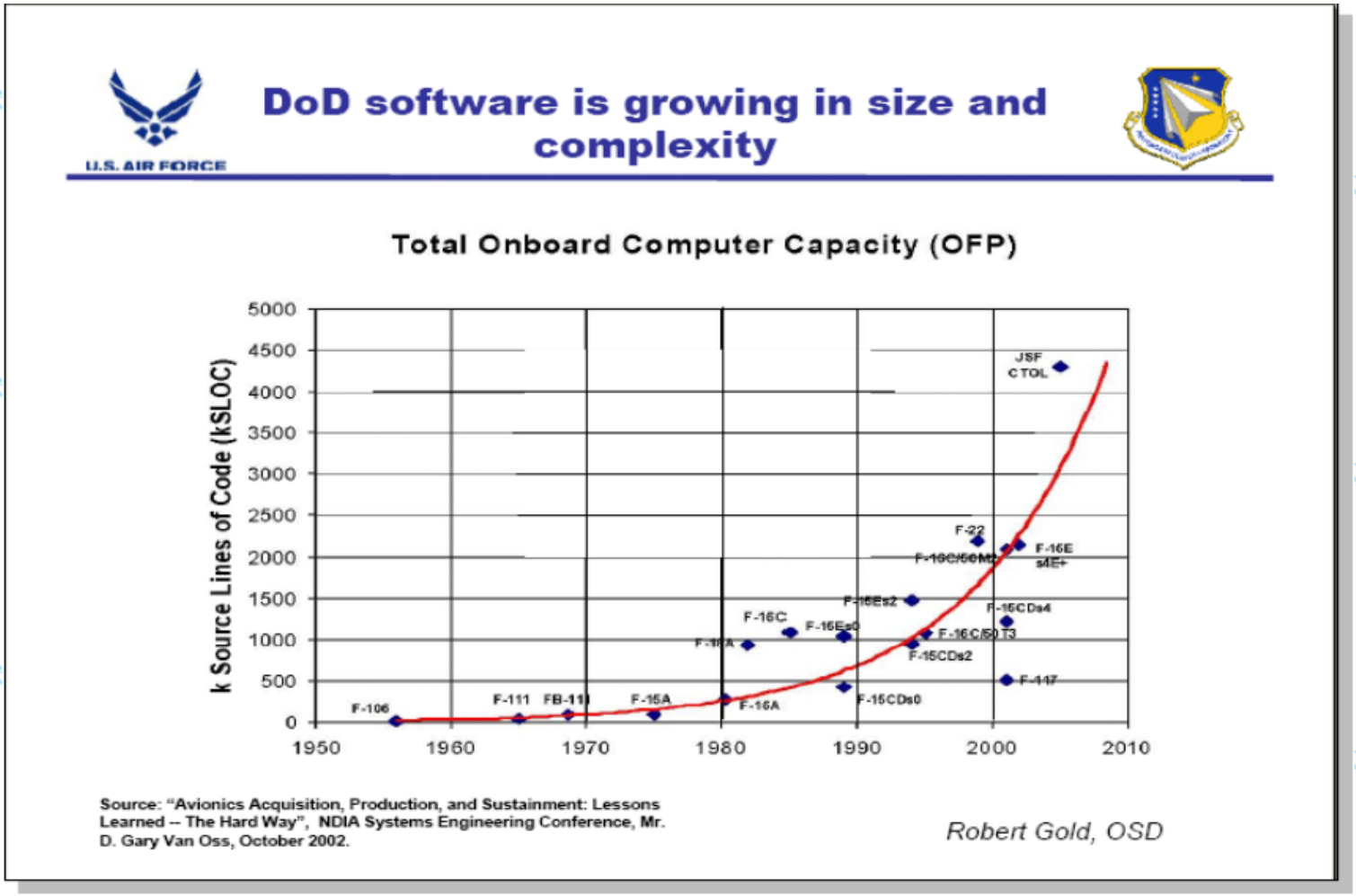
نرم‌افزار تعبیه‌شده

Software is now embedded everywhere Some of it is critical

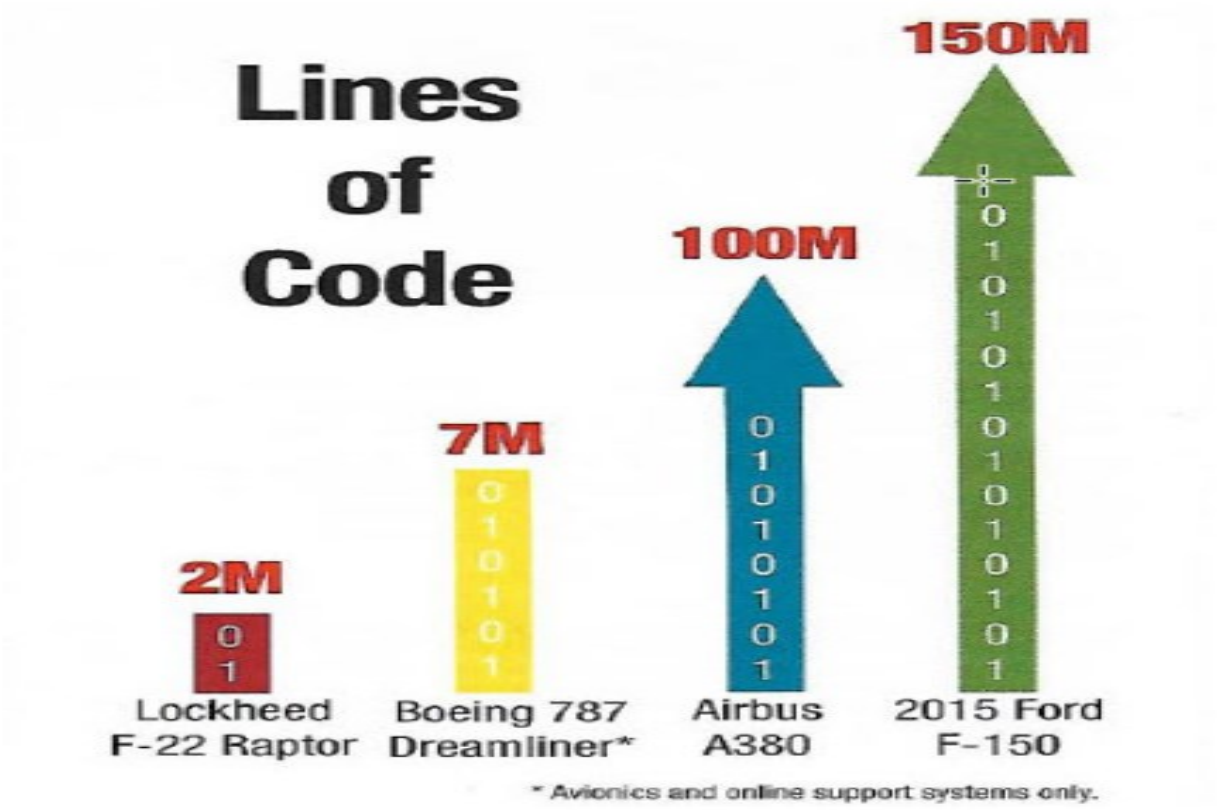


Failing software costs money and life!

افزایش سایز سیستم‌های نرم‌افزاری



افزایش سایز سیستم‌های نرم‌افزاری



افزایش سایز سیستم‌های نرم‌افزاری

Automotive Software

- A typical 2017 car model contains ~100M lines of code: how do you verify that?
- Current cars admit hundreds of onboard functions: how do you cover their combination?

E.g., does braking when changing the radio station and starting the windscreen wiper, affect air conditioning?

هزینه ناشی از خرابی نرم‌افزار

- Expensive recalls products with embedded software
- Lawsuits for loss of life or property damage
  - Car crashes (e.g., Toyota Camry 2005)
- Thousands of dollars for each minute of down-time
  - (e.g., Denver Airport Luggage Handling System)
- Huge losses of monetary and intellectual investment
  - Rocket boost failure (e.g., Ariane 5)
- Business failures associated with buggy software
  - (e.g., Ashton-Tate dBase)

هزینه ناشی از خرابی نرم‌افزار

- Potential problems are obvious:
  - Software used to control nuclear power plants
  - Air-traffic control systems
  - Spacecraft launch vehicle control
  - Embedded software in cars
- A well-known and tragic example: Therac-25 radiation machine failures

ویژگی سیستم‌های نرم‌افزاری

Software seems particularly prone to faults

Tiny faults can have catastrophic consequences

- Ariane 5
- Mars Climate Orbiter, Mars Sojourner
- Pentium-Bug
- . . .

Rare bugs can occur

- avg. lifetime of a passenger plane: 30 years
- avg. lifetime of a car: < 10 years, but already > 1:2B cars in 2014

Logic and implementation errors represent security exploits

(too many to mention)

ملاحظه

Building software is what most of you will do after graduation

- You'll be developing systems in the context above
- Given the increasing importance of software,
  - you may be liable for errors
  - your job may depend on your ability to produce reliable systems

What are the challenges in building reliable and secure software?

دستیابی به قابلیت اطمینان در مهندسی

Some well-known strategies from civil engineering:

- Precise calculations/estimations of forces, stress, etc.
- Hardware redundancy (“make it a bit stronger than necessary”)
- Robust design (single fault not catastrophic)
- Clear separation of subsystems (any airplane flies with dozens of known and minor defects)
- Design follows patterns that are proven to work

ویژگی متمایز برای سیستم‌های نرم افزار

- Software systems compute non-continuous functions
  Single bit-flip may change behavior completely
- Redundancy as replication doesn't help against bugs
  Redundant SW development only viable in extreme cases
- No physical or modal separation of subsystems
  Local failures often affect whole system
- Software designs have very high logic complexity
- Most SW engineers untrained in correctness
- Cost efficiency more important than reliability
- Design practice for reliable software is not yet mature

چگونه از صحت نرم افزار اطمینان حاصل کنیم؟

A Central Strategy: **Testing**
(others: SW processes, reviews, libraries, . . .)

Testing against inherent SW errors (“bugs”)

- Design test configurations that hopefully are representative and
- ensure that the system behaves as intended on them

Testing against external faults

- Inject faults (memory, communication) by simulation or radiation

محدودیت‌های آزمایش

- Testing can show the presence of errors, but not their *absence* (exhaustive testing viable only for trivial systems)

- *Representativeness* of test cases/injected faults is subjective
  How to test for the unexpected? Rare cases?

- Testing is labor intensive, hence expensive

آزمایش تکمیلی: تأیید رسمی

A Sorting Program:

```
int* sort(int* a) -
  ...
"
```

Testing sort:

- $\cdot \text{sort}(-3, 2, 5'') == -2, 3, 5''$ ✓
- $\cdot \text{sort}(-'') == -''$ ✓
- $\text{sort}(-17'') == -17''$ ✓

Typically missed test cases

- $\cdot \text{sort}(-2, 1, 2'') == -1, 2, 2''$ ☒
- $\cdot \text{sort}(\text{null}) == \text{exception}$ ☒
- $\cdot \text{isPermutation}(\text{sort}(a), a)$ ☒

تأیید رسمی به عنوان اثبات قضیه

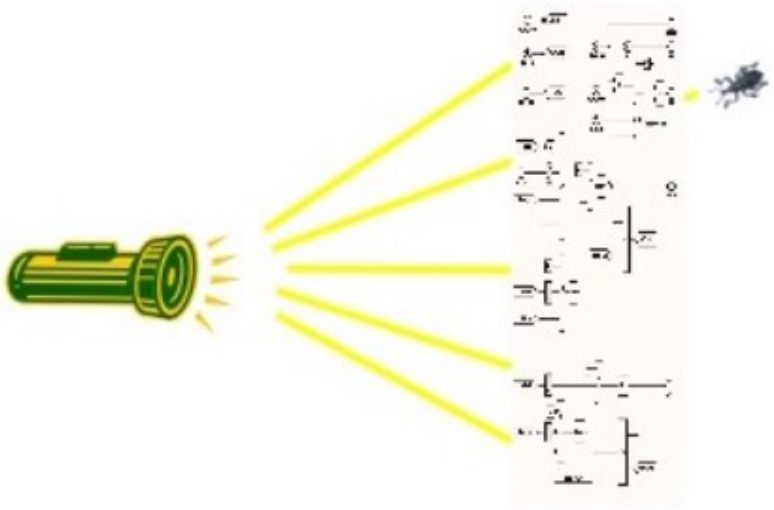
Theorem (Correctness of sort()**)**
For any given non-null int array a, calling the program sort(a) returns an int array that is sorted wrt $\leq$ *and is a permutation of* a.

However, methodology differs from mathematics:

1. Formalize the expected property in a logical language
2. Prove the property with the help of an (semi-)automatic tool

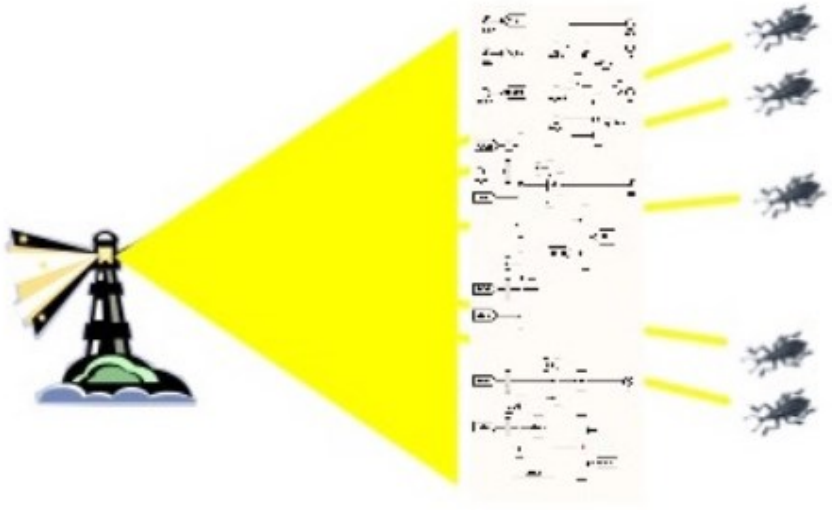
آزمایش متضاد با تأیید رسمی

Testing Checks Only the Values We Select



Even Small Systems Have Trillions (of Trillions) of Possible Tests!

Formal Verification Checks Every Possible Value!



Finds every exception to the property being checked!

روش‌های رسمی

Rigorous techniques and tools for the development and analysis of computational (hardware/software) systems

- Applied at various stages of the development cycle
- Also used in reverse engineering to model and analyze existing systems
- Based on mathematics and symbolic logic (formal)

موجودیت‌های اصلی در روش‌های رسمی

1. System requirements
2. System implementation

Formal methods rely on

a. some formal specification of (1)
b. some formal execution model of (2)

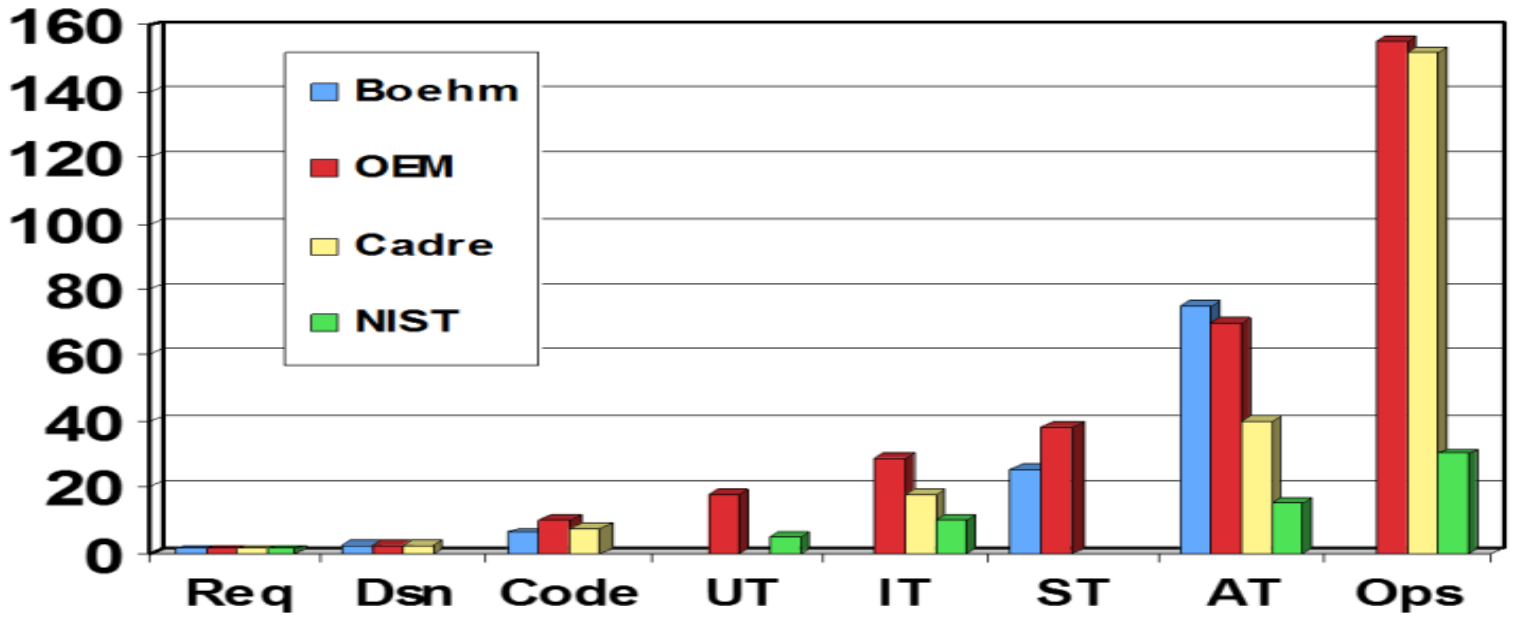
Use tools to verify mechanically that implementation satisfies (a) according to (b)

چرا از استفاده از روش‌های رسمی

- Mathematical modeling and analysis contribute to the overall quality of the final product
- Increase confidence in the correctness/robustness/security of a system
- Find more flaws and earlier (i.e., during specification and design vs. testing and maintenance)

چرا از استفاده از روش‌های رسمی

Relative cost to fix an error, by development phase



Finding errors earlier reduces development costs

روسهای رسمی: چشم انداز

- Complement other analysis and design methods
- Help find bugs in code and specification
- Reduce development, and testing, cost
- Ensure certain properties of the formal system model
- Should be highly automated

روش‌های رسمی و آزمایش

- Run the system at chosen inputs and observe its behavior
  - Randomly chosen
  - Intelligently chosen (by hand: expensive!)
  - Automatically chosen (need formalized spec)
- What about other inputs? (test coverage)
- What about the observation? (test oracle)

Challenges can be addressed by/require formal methods

اخطار

- The notion of “formality” is often misunderstood (formal vs. rigorous)
- The effectiveness of formal methods is still debated
- There are persistent myths about their practicality and cost
- Formal methods are not yet widespread in industry
- They are mostly used in the development of safety, business, or mission critical software, where the cost of faults is high

نکته اصلی روشهای رسمی این نیست

- To show “correctness” of entire systems
  - What is correctness? Go for specific properties!
- To replace testing entirely
  - Formal methods do not go below byte code level
  - Some properties are not formalizable
- To replace good design practices

There is no silver bullet!
No correct system w/o clear requirements & good design

مزایای کلی استفاده از روش های رسمی

- Forces developers to think systematically about issues
- Improves the quality of specifications, even without formal verification
- Leads to better design
- Provides a precise reference to check requirements against
- Provides documentation within a team of developers
- Gives direction to latter development phases
- Provides a basis for reuse via specification matching
- Can replace (infinitely) many test cases
- Facilitates automatic test case generation

مشخصات: آنچه سیستم باید انجام دهد

- Simple properties
  - Safety properties: something bad will never happen
  - Liveness properties: something good will happen eventually
  - Non-functional properties: runtime, memory, usability, . . .

- “Complete” behaviour specification
  - Equivalence check
  - Refinement
  - Data consistency
  - . . .

مشخصات رسمی

The expression in some formal language and at some level of abstraction of a collection of properties that some system should satisfy [van Lamsweerde]

- formal language:
  - syntax can be mechanically processed and checked
  - semantics is defined unambiguously by mathematical means
- abstraction:
  - above the level of source code
  - several levels possible

مشخصات رسمی

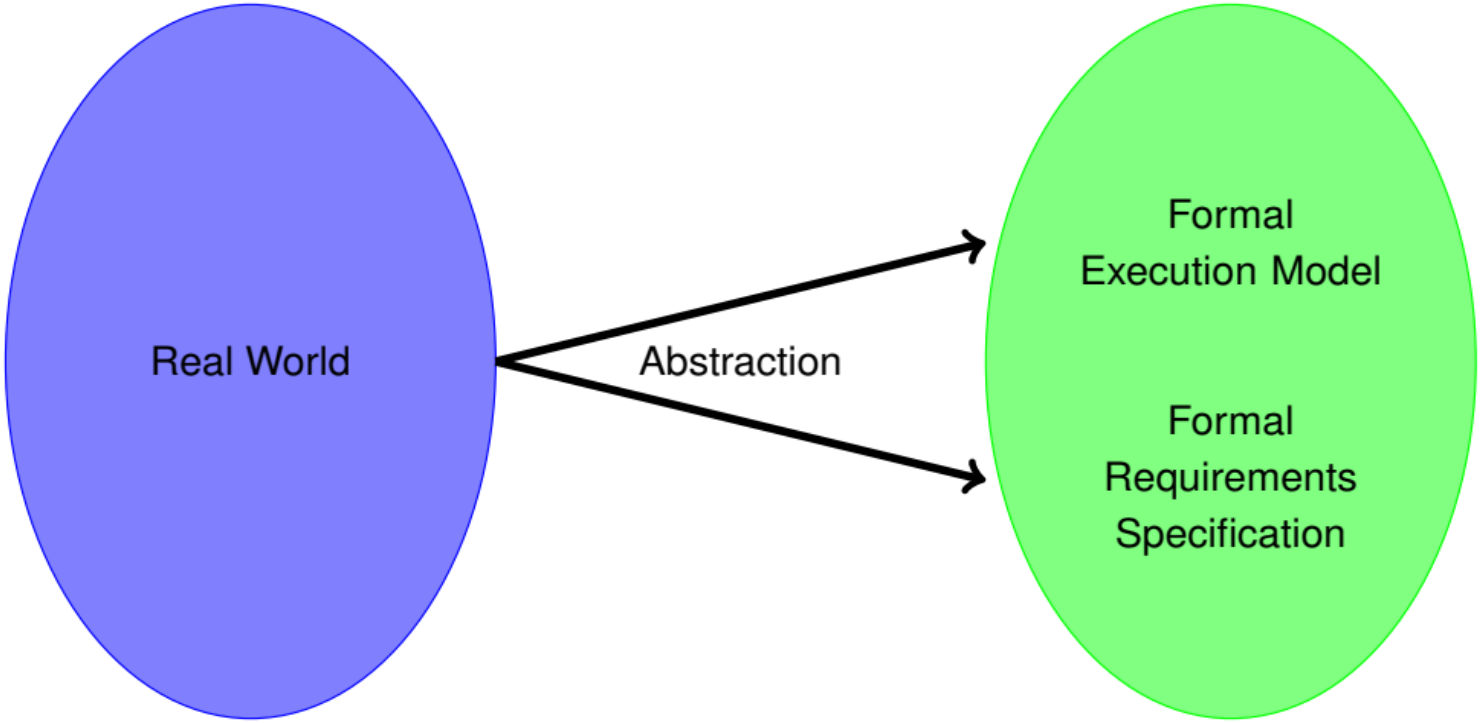
The expression in some formal language and at some level of abstraction of a collection of properties that some system should satisfy [van Lamsweerde]

- properties:
  - expressed in some formal logic
  - have a well-defined semantics
- satisfaction:
  - ideally (but not always) decided mechanically
  - based on automated deduction and/or model checking techniques

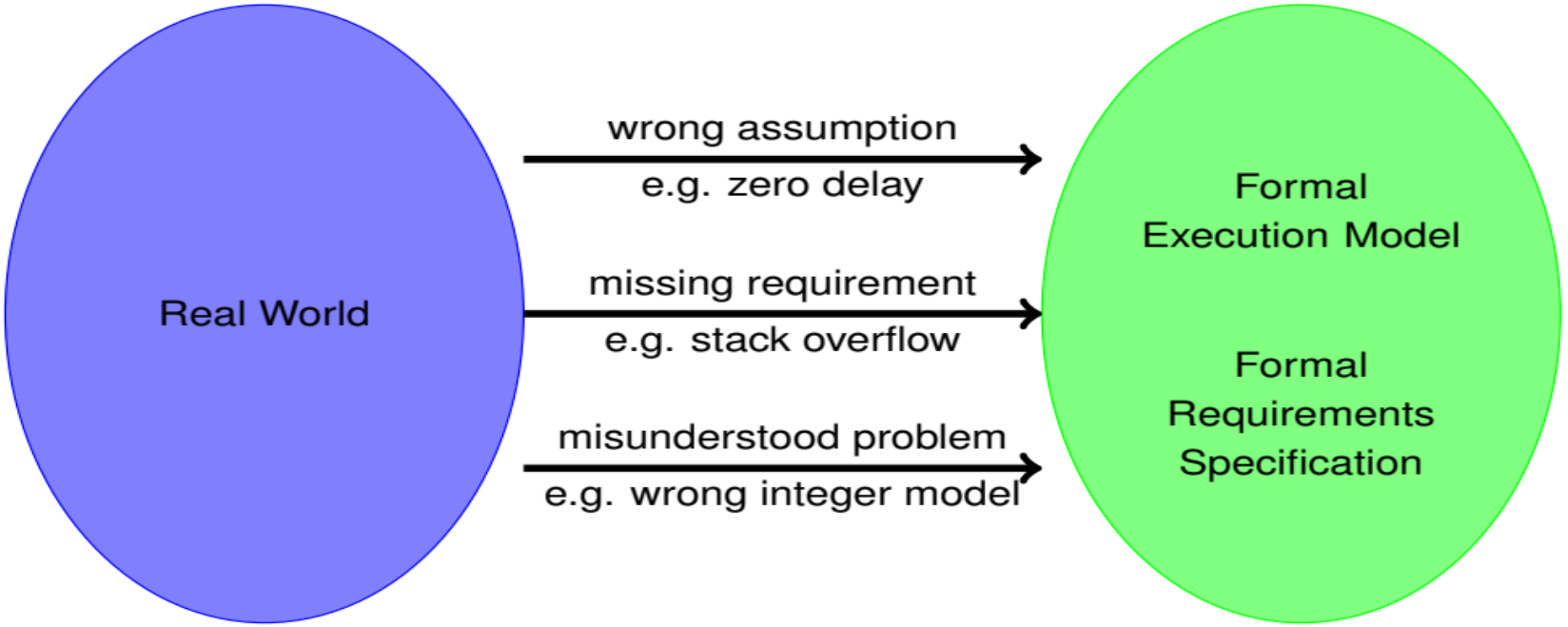
واقعیت اساسی

- Formalisation of system requirements is hard

مشکلات ایجاد مدل‌های رسمی



مشکلات ایجاد مدل‌های رسمی



واقعیت اساسی دیگر

- Proving properties of systems can be hard

سطح توصیف سیستم / بیان مشخصات

High level (modeling/programming language level)

- Complex datatypes and control structures, general programs
  - General properties
- Easier to program
  - High precision, tight modeling
- Automatic proofs (in general) impossible!

Low level (machine level)

- Finitely many states
  - Finitely many cases
- Tedious to program, worse to maintain
  - Approximation, low precision
- Automatic proofs are (in principle) possible

روندهای فعلی و آینده

Slowly but surely formal methods are finding increased used in industry.

- Designing for formal verification
- Combining semi-automatic methods with SAT/SMT solvers, theorem provers
- Combining static analysis of programs with automatic methods and with theorem provers
- Combining test and formal verification
- Integration of formal methods into SW development process

روندهای فعلی و آینده

Need for secure systems is increasing the use of FMs

- Security is intrinsically hard
- Redundant fault-tolerant systems are often used to meet safety requirements
- Fault-tolerance depends on the independence of component failures
- Security attacks are intelligent, coordinated and malicious
- Formal methods provides a systematic way to meet strict security requirements

خلاصه

- Software is becoming pervasive and very complex
- Current development techniques are inadequate
- Formal methods . . .
  - are not a panacea, but will be increasingly necessary
  - are (more and more) used in practice
  - can shorten development time
  - can push the limits of feasible complexity
  - can increase product quality
  - can improve system security
- We will learn to use several different formal methods, for different development stages

قدردانی

- Haniel Barbosa from uiowa university
- TA: Parisa Akhbari