Lecture 1: Introduction to Embedded System Verification CS/ECE584

September 4th 2012

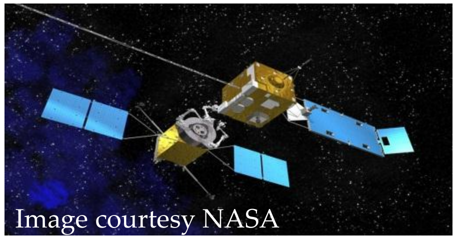
Sayan Mitra

Plan for today

- Administrivia
  - Project
- Background concepts
  - OR A Brief History of Mechaned Reasoning

Motivation for **Embedded** System Verification

- Examples of "Embedded" or "Cyber-Physical" Systems



- Characteristics
  - A control system
  - implemented in software
  - with many sensors, signal & data processing algorithms
  - communications over networks

dynamics

state machines

quantization

message drops

Motivation for **Embedded** System Verification

- Unlike "one-shot" function computations, computations of embedded systems are infinite streams
  - Example: Rudder positions computed by an autopilot program: L, R, R, R, C, L, …
- Testing and simulations (at least their naïve applications) can check for only finitely many behaviors
- Not sufficient for **covering** all behaviors of Embedded Systems

- Is this a real problem?
- Yes!

"**June 4, 1996 -- Ariane 5 Flight 501.** Working code for the Ariane 4 rocket is reused in the Ariane 5, but the Ariane 5's faster engines trigger a bug in an arithmetic routine inside the rocket's flight computer. The error is in the code that converts a 64-bit floating-point number to a 16-bit signed integer. The faster engines cause the 64-bit numbers to be larger in the Ariane 5 than in the Ariane 4, triggering an overflow condition that results in the flight computer crashing." --- *History's Worst Software Bugs, Simson Garfinkel, Wired 2005*

Motivation for Embedded System Verification

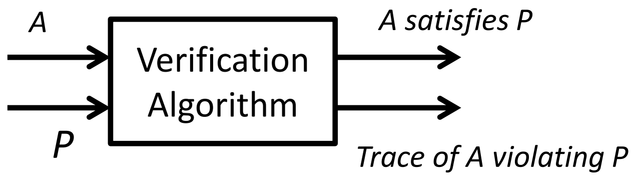
CalTech's autonomous vehicle ALICE disqualified in the 3rd round of the DARPA Urban challenge after an unforeseen interaction between the **path planner** and the **obstacle-avoidance** algorithm caused ALICE to crash through the road barriers

More Epic Fails in Embedded Systems:
Mariner I space probe crash, Chinook helicopter crashed (1994, killed all 29 passengers), Swedish JAS 39 Gripen fighter crash (1993), Mars climate orbiter crash ($125 M, 1999), Patriot missile time to float conversion error (28 soldiers), …
More general systems: Pentium Div ($475 M), Knight Capital Group's algorithmic trading software caused massive volume buys and sells ($440 M),

Motivation for Embedded System Verification

- The promise of Automatic Verification
- An algorithm takes as input
  - (a) the description of system *A* and
  - (b) property *P*

  and terminates with output
  - (c) a proof that all the behaviors of A satisfy P OR
  - (d) a particular behavior of A that violates P



- Examples:
  1. *A:* model of autonomous vehicle *P:* always stays on the road
  2. *A:* model of a traffic control system *P:* vehicles do not collide

Guarded Optimism

- Hardware verification (model checking) is now part of engineering practice in the industry
- Synchronous languages like Esterel and Lustre and their analysis suites are well-adopted in avionics and process control industries
- Success stories from Software Verification: SLAM tool from MSR
- Commercial and non-profit verification enterprises
  - Big EDA: Synopsis, Mentor Graphics, Cadence
  - Jasper, Coverity, Galois, SRI, etc.
- Explosive growth in academic research
  - International conferences: HSCC, EMSoft, ICCPS, CAV, TACAS…
- Plenty of room for research and entrepreneurship

Learning Objectives

- Techniques and formalisms for **modeling** systems with dynamics, computation, and communication
  - Hybrid Automata
- To **use verification tools** (model checkers, SMT solvers, and theorem provers)
- **Exposure** to some of the best ideas in CS, current research directions & trends
- Positive side-effects: connection to synthesis

Administrivia & Course Overview

- Tools: PVS, PHAVer, SpaceEx, UPPAAL
  - Download and install
  - Start looking at the tutorials and examples
- Webpage:http://engr-courses.engr.illinois.edu/ece584/index.shtml
- Mailing list: illinois-ece584@googlegroups.com
- No Exams!
- Homework (~40%): 3-5 sets with theoretical and programming problems
- Project (~60%): Project ideas have been announced
- Class participation (10%): Discussion, refine slides & notes

Brief History of Mechanized Reasoning

BACKGROUND

- Gottfried Liebniz (1646-1716) proposed the development of a “formal system” that would reduce proving validity of statements to calculations
  - Settle all controversy on any subject whatsoever could be settled by “taking their pens in their hands and calculating”
  - Reasoning without worrying about the veracity of individual propositions or statements

Digging deeper into Proofs

- What is a Proof?
- A **proof** is **sufficient evidence** for the truth of a proposition.
  - Real world and science: Evidence is drawn from nature and experiments
  - Law: Evidence comes from witnesses and forensics; “principle of beyond sufficient doubt”
  - A mathematical **proof** of a proposition is sequence of unambiguous statements that demonstrate its validity assuming some axioms

Example: Euclid’s Geometry

- Axioms
  1. A straight line segment can be drawn joining any two points.
  2. Any straight line segment can be extended indefinitely in a straight line.
  3. Given any straight line segment, a circle can be drawn having the segment as radius and one endpoint as center.
  4. All right angles are congruent.
  5. Given any straight line and a point not on it, there "exists one and only one straight line which passes" through that point and never intersects the first line, no matter how far they are extended.
- Theorems: Statements about objects in plane geometr
  - Sum of angles of a triangle equals 180 degrees
  - Pythagoras’ Theorem
  - …

Propositional Logic

- **Syntax** (rules for constructing well formed sentences)
  - Countable set of (atomic) propositions PS: P1, P2, P3, …
  - S = $\text{True} \mid \text{False} \mid p_1 \mid \neg p_1 \mid p_1 \land p_2 \mid p_1 \lor p_2 \mid p_1 \Rightarrow p_2 \mid p_1 \equiv p_2 \mid$ (S), where p_1, p_2 are variables of type

- Example: PS = A, B, C then the following are well-formed propositional statements
  - A, (A ∧ B), (A ∧ B) ⇒ (C ∨ A), True ⇒ A, …

Semantics of Propositional Logic

- Let PROPS be the set of all possible propositional logic statements
- Semantics defines a truth value functions or **valuations** v that maps each proposition PS to a truth value (T or F), v: PS→ {T, F} and by extension a valuation v':PROPS→{T,F}
  - The valuation of a statement is inductively defined by the valuation of the propositions and the truth table of the operators
  - Example (cont.): if v(A) = T, v(B) = T, v(C)= F then A, v'(A ∧ B) = T, v'((A ∧ B) ⇒ (C ∨ A)) = T, v'(True ⇒ A) = T, …

p_1	p_2	$\neg p_2$	$p_1 \land p_2$	$p_1 \Rightarrow p_2$	$p_1 \equiv p_2$
F	F	T	F	T	T
F	T	F	F	T	F
T	F	T	F	F	F
T	T	F	T	T	T

Satisfiability, Validity, and Tautologies

- A proposition A is **valid** v'(A) = T for all valuations v. A is also called a **tautology**
- Example: A ⇒ A, A ∨ ¬ A, (A ⇒ B) ≡ (¬ B ⇒ ¬A)
- A proposition is **satisfiable** if there is a valuation (or truth assignment) v such that v(A) = T.
- Example (P ∨ Q) /\ (¬ P ∨ ¬ Q) is satisfied by the valuation v(P) = F and v(Q) = T
- A proposition is **unsatisfiable** if it is not satisfied by any valuation
- (P ∨ Q) /\ (¬ P ∨ ¬ Q) /\ P is unsatisfiable
- **Lemma.** A proposition A is a valid if and only if ¬A is unsatisfiable.
- Checking (un)satisfiability is called **boolean satisfiability problem** (SAT).
- How to check that a given proposition is valid / satisfiable?
  - Truth table method (check all possible 2^n valuations)
  - **Decision Procedure**: An algorithm for solving a decision problem (e.g. SAT, Validity)
  - Unfortunately, the above exponential search is also the best we can do, in theory. SAT is **NP-complete**
  - Though modern SAT solvers routinely handle propositions with millions of atomic propositions and constraints

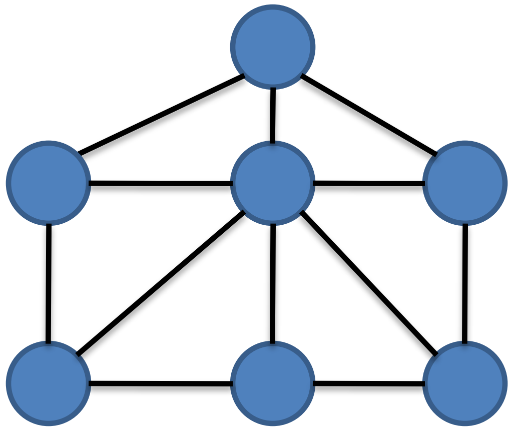
Four colors suffice

- Any 2D map can be colored with 4 colors
- Any **planar** graph can be colored with 4 colors!
- This is the (famous) 4 color theorem proposed in 1852 when Francis Guthrie (to De Morgan), while trying to color the map of counties of England

Washington
Oregon
California
Idaho
Nevada
Montana
Wyoming
Utah
Colorado
Arizona
New Mexico
North Dakota
South Dakota
Nebraska
Kansas
Oklahoma
Texas
Minnesota
Iowa
Missouri
Arkansas
Louisiana
Wisconsin
Illinois
Indiana
Michigan
Ohio
Kentucky
Tennessee
Mississippi
Alabama
Georgia
Florida
South Carolina
North Carolina
Virginia
West Virginia
Pennsylvania
New York
Maine
New Hampshire
Vermont
Massachusetts
Rhode Island
Connecticut
New Jersey
Delaware
Maryland
Washington, D.C.
Alaska
Hawaii

Kenneth Appel and Wolfgang Haken (1976, at UIUC) proved the four color theorem to much acclaim!

Proof reduced infinite set of possible maps to 1,936 reducible configurations which had to be checked one by one by computer and took > 1000 hours



Revisiting Definition of a Proof

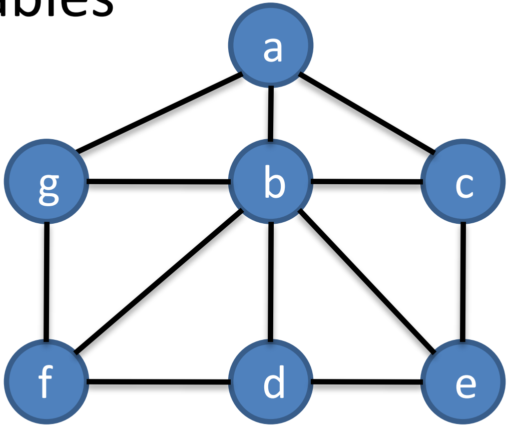
- A mathematical proof of a proposition is sequence of unambiguous statements that demonstrate **(to whom?)** its validity assuming some axioms

- Human mathematicians?
- Computers?

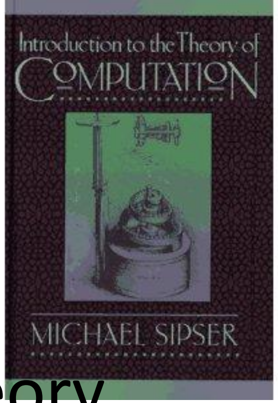
- Proof of the 4-color theorem

Graph/Map Coloring with SAT

- For each of the k vertices create 2 binary variables
  - a_{00} = T iff vertex a is RED
  - a_{01} = T iff vertex a is BLUE
  - a_{10} = T iff vertex a is YELLOW
  - a_{11} = T iff vertex a is GREEN
- Encode the graph constraints
  - $a_{00} \Rightarrow (\neg a_{01} \wedge \neg a_{10} \wedge \neg a_{11} \wedge) \wedge \ldots$
  - $(\neg (a_{00} \equiv b_{00}) \vee \neg (a_{01} \equiv b_{01}) \vee \neg (a_{10} \equiv b_{10}) \vee \neg (a_{11} \equiv b_{11})) \wedge \ldots$
- See this page http://www.cs.cmu.edu/~bryant/boolean/macgregor.html



Reading Assignments

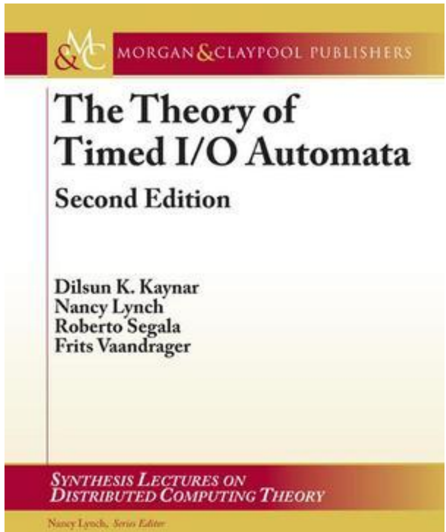


Introduction to the Theory of COMPUTATION

MICHAEL SIPSER

- Further reading: any standard textbook on theory of computation, e.g., Introduction to Theory of Computation by Michael Sipser
  - Turing Machines
  - Decidability
  - Complexity classes P, NP, NP-hard, NP-complete
- Next: Predicate Logic, and

  Timed Automata



M&C MORGAN & CLAYPOOL PUBLISHERS

The Theory of Timed I/O Automata

Second Edition

Dilsun K. Kaynar
Nancy Lynch
Roberto Segala
Frits Vaandrager

SYNTHESIS LECTURES ON DISTRIBUTED COMPUTING THEORY

Nancy Lynch, Series Editor