Formal Design and Analysis of Cyber-Physical Systems

Rajeev Alur

University of Pennsylvania
Embedded Software Systems Everywhere!
From Desktops to Cyber-Physical Systems

- Traditional computers: Stand-alone device running software applications (e.g. data processing)

- Traditional controllers: Devices interacting with physical world via sensors and actuators (e.g. thermostat)

- Embedded Systems
  - Special-purpose system with integrated microcontroller/software
  - Cameras, watches, washing machines...
Cyber-Physical Systems

Control
Monitor and influence physical world

Computation
Process information to make decisions

Communication
Exchange data to collaborate
Cyber-Physical Systems

Driverless Cars

Medical devices

Coordinating robots
Cyber-Physical Systems

Systems that integrate control, computation, and communication can do
Cool things,
And useful things...
Lots of promise and potential: medicine, transportation, energy, ...
So what’s the challenge?
Ariane 5 Explosion

It took the European Space Agency 10 years and $7 billion to produce Ariane 5. All it took to explode that rocket less than a minute into its maiden voyage last June, scattering fiery rubble across the mangrove swamps of French Guiana, was a small computer program trying to stuff a 64-bit number into a 16-bit space.

Toyota officials described the problem as a "disconnect" in the vehicle's complex anti-lock brake system (ABS) that causes less than a one-second lag. With the delay, a vehicle going 60 mph will have traveled nearly another 90 feet before the brakes begin to take hold.
Software: The Achilles’ Heel

Software everywhere means bugs everywhere

Study by NIST:

Software bugs cost US economy $60 billion annually (0.6% of GDP)

Lack of trust in software as technology barrier

Would you use an autonomous software-controlled round-the-clock monitoring and drug-delivery device?
Johnson & Johnson is telling patients that it has learned of a security vulnerability in one of its insulin pumps that a hacker could exploit to overdose diabetic patients with insulin, though it describes the risk as low.

Reuters Oct 4, 2016
A Plausible Solution for High-Assurance Systems

- Model-based Design
- Formal Verification
Model-based design != Coding

Design using high-level block diagrams and state machines gets automatically compiled into low-level code!

Models not only of system being designed, but also of its environment.
Program testing can be used to show the presence of bugs, but never their absence!

Edsger W. Dijkstra
Formal Verification

- **Goal**: Establish that model satisfies requirements under all possible scenarios

- **First challenge**: Need formal definitions of “model” and “requirement” to make the problem mathematically precise

- **Second challenge**: Need verification techniques and tools
Case Study: Implantable Pacemaker

Goal: Monitor timing pattern of heartbeats and correct irregularities using external electrical stimuli

Detecting and correcting Bradycardia using DDD pacemaker
First step in model-based design:
Identify components/processes and communication interface

Sensing events: AI, VI from leads in atrium and ventricle

Pacing events: AP, VP sending electrical stimuli from pacemaker
Decomposition into Sub-components

FilterA/FilterV remove noise from input signals

PaceA/PaceV produce pacing events to maintain minimum heart rate
Specifying Behavior: Atrial Pacing

Timed automata:
Finite-state machine + Clock variables to express bounds on delays

If no atrial sensing event (AS) for a duration of LRI (Lower Rate Interval) since the most recent ventricular event (VP or VS), then output the pacing event VP
Specifying Behavior: Atrial Pacing

Clock variable $x$ reset to 0 on every ventricular event
As time elapses, value of $x$ increases reflecting time spent in Pending

Invariant condition ($x < \text{LRI}$) for Pending and transition guard ($x = \text{LRI}$) ensure desired timing constraint on generation of AP event
A Simple Heart model

Atrial and ventricular events are generated independently

Generate atrial sensing event AI after a nondeterministic delay in the interval $[L, U]$ since the previous atrial event
Formalizing Requirements

Safety Monitor:
Observes input/output behavior of the system, and enters an error state if an undesirable behavior is detected

Analysis/verification problem:
Given a system and a monitor, is there an execution of the system that leads the monitor into its error state?
Basic safety for Bradycardia:
Maintain ventricular rate above threshold of $R$

Monitor enters error state if it detects a violation of safety requirement
Verification problem:
Is there an execution of system that leads monitor to error?

Model checking:
Explore all executions / reachable states using symbolic methods

Uppaal:
Efficient model checker for timed automata (www.uppaal.org)
Summary of Verification Results

- Modeled several features of pacemaker algorithm from Boston Scientific
- Verified against requirements published by FDA
- Design exploration
  - Which parameter combinations maintain safety?
  - Exploring variants of detection strategies
  - Adding new features

Reference: Closed loop verification of medical devices with abstraction and refinement, Jiang et al, STTT, 2014
Formal Design and Analysis of CPS

- **Model-based Design**
  - Block diagrams, state machines, hierarchy, timing constraints
  - Formal semantics offers basis for code generation / analysis
  - Commercial tools: Stateflow/Simulink, Modelica, …
  - Academic tools: Charon (Penn), Ptolemy (UC Berkeley), …

- **Formal verification**
  - Need formal requirements
  - Industry norm: advanced simulation (Simulink Design Verifier)
  - Academic tools: Uppaal, SpaceEx, Taliro, …

- **Application domains**
  - Medical devices (pacemaker, insulin pump, …)
  - Automotive software (collision avoidance, …)
  - Smart buildings (energy efficient control, …)
Challenge: Certified CPS Software

Can FDA use formal analysis tools to certify medical software/devices?
How to verify actual implementation? (code + hardware)

Opportunity:
Can heart model and diagnosis be patient specific based on data?
Challenge: Formal Methods for Security

How to formalize security and protection against attacks?

Relatively well understood and studied aspects:
  Encryption schemes and key-exchange protocols
  Information flow analysis to detect leakage of confidential data

Research opportunities
  Side channel attacks (e.g. timing attacks)

How to account for unexpected threats at design time?
Challenge: Analysis of Machine-Learning-based CPS

How can we verify an autonomous car?

Challenge: no suitable high-level model of decisions made by systems relying on machine learning algorithms

Opportunity: Understandable/explainable AI