Model Synthesis

New Challenges in Model Based Design

Rajeev Alur
University of Pennsylvania

Medical CPS NSF Site Visit, Jan 2012

Software: Key to Embedded Revolution

Software Inside!

Software ⇒ New features, Automation, Customization

Software Bugs, Unpredictability, Recalls
Prius Brake Problems Blamed on Software Glitches

“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” (source: CNN Feb 4, 2010)

Embedded Computation

- Typical embedded program: cruise control
  Loop
  - Read the sensors;
  - Compute speed;
  - Compute pressure for brake pedal / accelerator;
  - Transmit the outputs to actuators;

- Program has (non-terminating) interaction with the outside world: Reactive computation

- Correctness depends on real-time response (does the car brake fast enough?)

- Analysis of correctness requires modeling of the dynamics of the car
From 1985-2005, nearly 30,000 deaths and 600,000 injuries from device failures

From 1996-2006, the percentage of software-related causes in medical device recalls have grown from 10% to 21% (Complexity ↑ → Potential safety violations ↑)

There is currently no well-established standards for development of software for medical devices

Model-based Pacemaker Software Design

Heart | Pacemaker | Verification | Simulation | Testing
---|---|---|---|---
Random Heart Model | UPPAAL Model
Virtual Heart Model | Simulink Model
Heart Model Implementation | Pacemaker Implementation
Model Verification

Heart

Random Heart Model

Virtual Heart Model

Heart Model Implementation

Pacemaker

UPPAAL Model

UPPAAL: A tool for verification of network of Timed-automata

Testing

Jiang, Pajic, Moarref, Alur, Mangharam, TACAS 2012

Model Simulation

Heart

Random Heart Model

Virtual Heart Model

Heart Model Implementation

Pacemaker

UPPAAL Model

Simulink Model

Timing-based heart model and interface in Simulink

Testing

Jiang, Pajic, Mangharam, ICCPS, Proc. IEEE, 2011
Model Translation and Implementation

Heart
Random Heart Model
Virtual Heart Model
Heart Model Implementation

Pacemaker
UPPAAL Model
Simulink Model
Pacemaker Implementation

Verification
Simulation
Testing

Pajic, Jiang, Lee, Mangharam, Sokolsky, RTAS 2012

Implantable Pacemaker Modeling
**Uppaal Model of Dual Chamber Pacemaker**

(a) LRI component  
(b) AVI component  
(c) URI component  
(d) PVARP component  
(e) VRP component

**Summary of Verification Results**

- Modeled and verified a dual chamber pacemaker and additional advanced functions.
- Showed that adding new functions to the pacemaker may result in safety violations.
- Showed that more detailed heart model is needed for more advanced safety requirements.
New Challenges: Model Synthesis

1. Can we extract (controller) models from code?

2. Can we extract (plant) models from data?

3. Can we use analysis/verification technology to assist the designer to construct models?

1. From Code to Models

- What if pacemaker software is not developed using MBD
  - How to verify / certify code for pacemaker software

- Potential solution: Extract EFSM (Extended finite-state-machine) models from code

- Starting point: Predicate abstraction used for software verification
  - Challenges: Extract timing properties
Model Checking of C code

Phase 1: Given a program P, build an abstract finite-state (Boolean) model A such that set of behaviors of P is a subset of those of A (conservative abstraction)

Phase 2: Model check A wrt specification: this can prove P to be correct, or reveal a bug in P, or suggest inadequacy of A

- Shown to be effective on Windows device drivers in Microsoft Research project SLAM

```c
do{
    KeAcquireSpinLock();
    nPacketsOld = nPackets;
    if(request){
        request = request->Next;
        KeReleaseSpinLock();
        nPackets++;
    }
}while(nPackets!=nPacketsOld);
KeReleaseSpinLock();
```

Do lock operations, acquire and release, strictly alternate on every program execution?

---

2. From Data to Models

- What's a good model of heart for verifying correctness requirements of pacemaker software?
  - Ideally, model should be patient specific

- Potential solution: Extract timed/hybrid automata models for ECG data for a patient

- Computational challenge: Can we develop suitable learning algorithms?
  - Background: L* algorithm for learning DFA
  - Background: Learning linear constraints among variables
3. Assisting Designers in Model Construction

- How can computational tools assist designers?
  - Maturing verification technology, fast constraint solvers
  - Enormous computational power available

- Goal: Allow designer to express “model under construction” using multiple, intuitive formats
  - Synthesis tool can integrate different formats, interactively with designer to produce desired model
  - EFSMs + Example scenarios + High-level requirements

- Inspiration: Emerging research in software synthesis

Sketch: Program completion

Ref: Chaudhuri, Solar-Lezama (PLDI 2010)

```plaintext
Err = 0.0;
for(t = 0; t<T; t+=dT){
    if(stage==STRAIGHT){
        if(t > ??) stage= INTURN;
    }
    if(stage==INTURN){
        car.ang = car.ang < ??;
        if(t > ??) stage= OUTTURN;
    }
    if(stage==OUTTURN){
        car.ang = car.ang + ??;
        if(t > ??) break;
    }
    simulate_car(car);
    Err += check_collision(car);
    Err += check_destination(car);
}
```

When to start turning?  
Backup straight  
How much to turn?  
Straighten  

Enables programmers to focus on high-level solution strategy
Conclusions

- Model based design (MBD) is a promising approach to design of embedded software

- Over the past year, research at Penn has demonstrated benefits of MBD for rigorous design of pacemaker software

- Synthesis has the potential to transform the way a designer can employ MBD