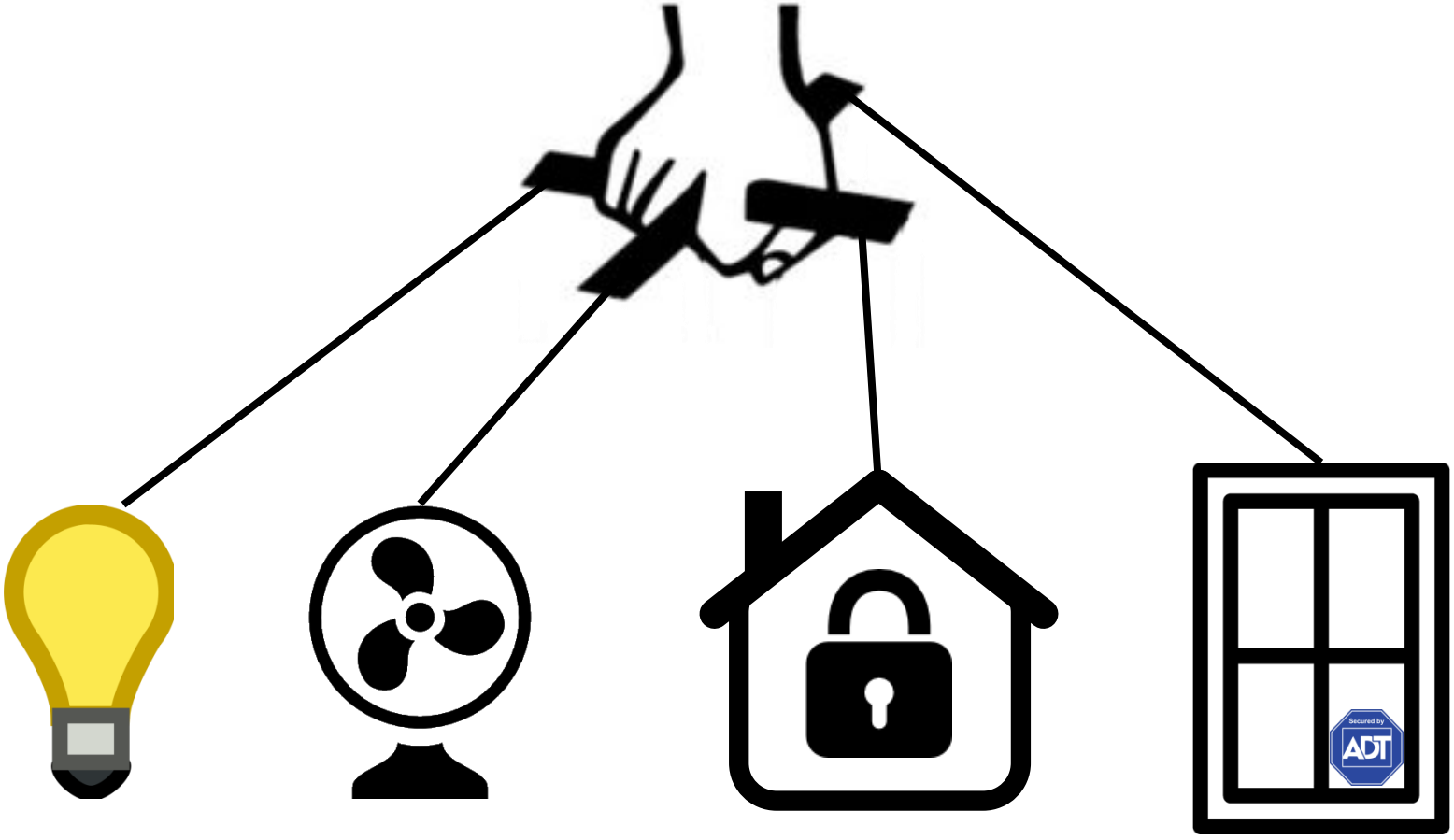
Systematically Exploring the Behavior of Control Programs

Jason Croft, Ratul Mahajan,
Matthew Caesar, Madan Musuvathi

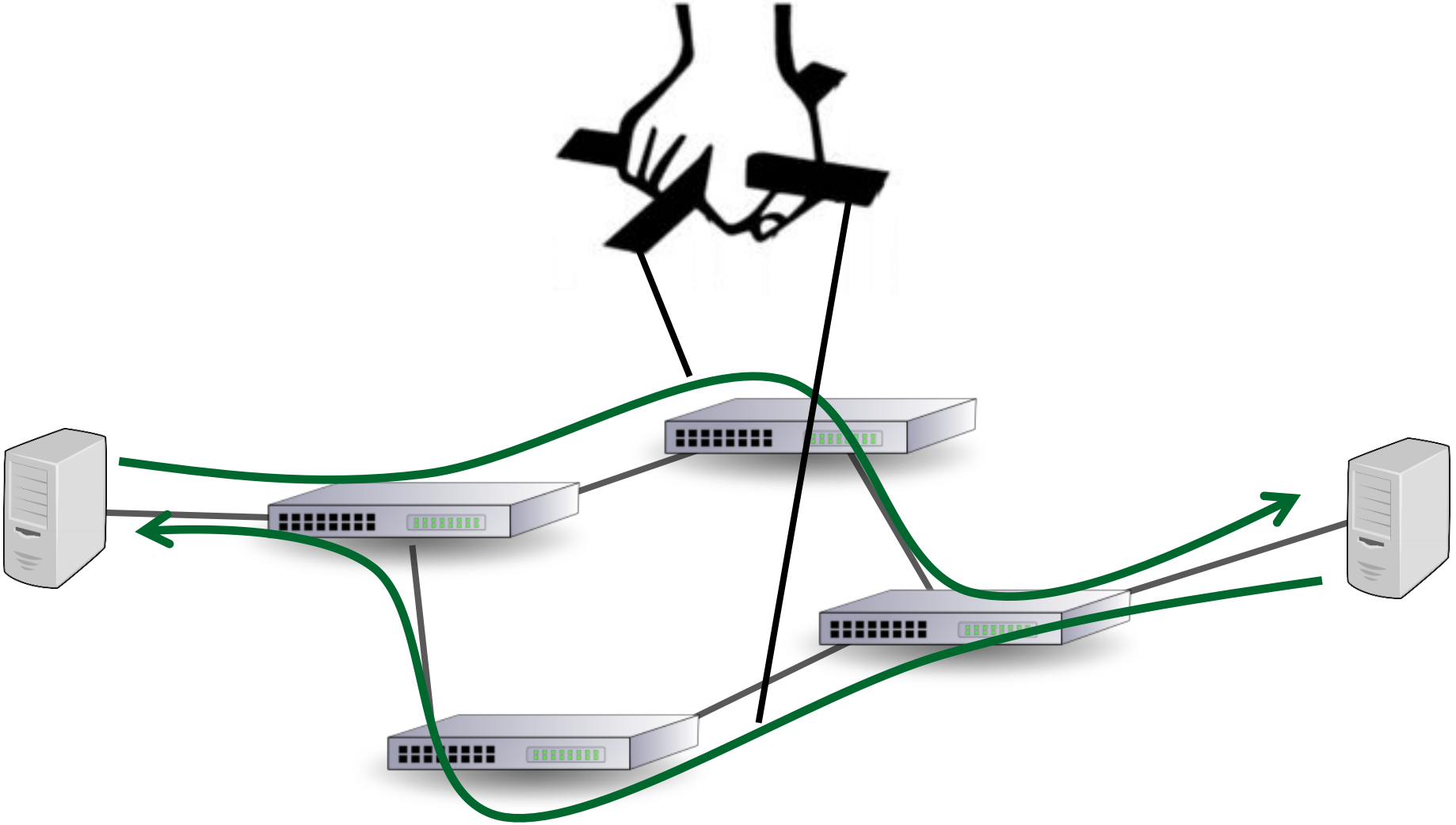
ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Microsoft Research

Control Programs are Everywhere



Control Programs are Everywhere



Control Programs are Everywhere

Triggers

Actions

```
motionPorch.Detected:
    if (Now – timeLastMotion < 1s
        && lightLevel < 20)
           porchLight.Set(On)
    timeLastMotion = Now

@6:00:00 AM:
    porchLight.Set(Off)

@6:00:00 PM:
    porchLight.Set(On)
```

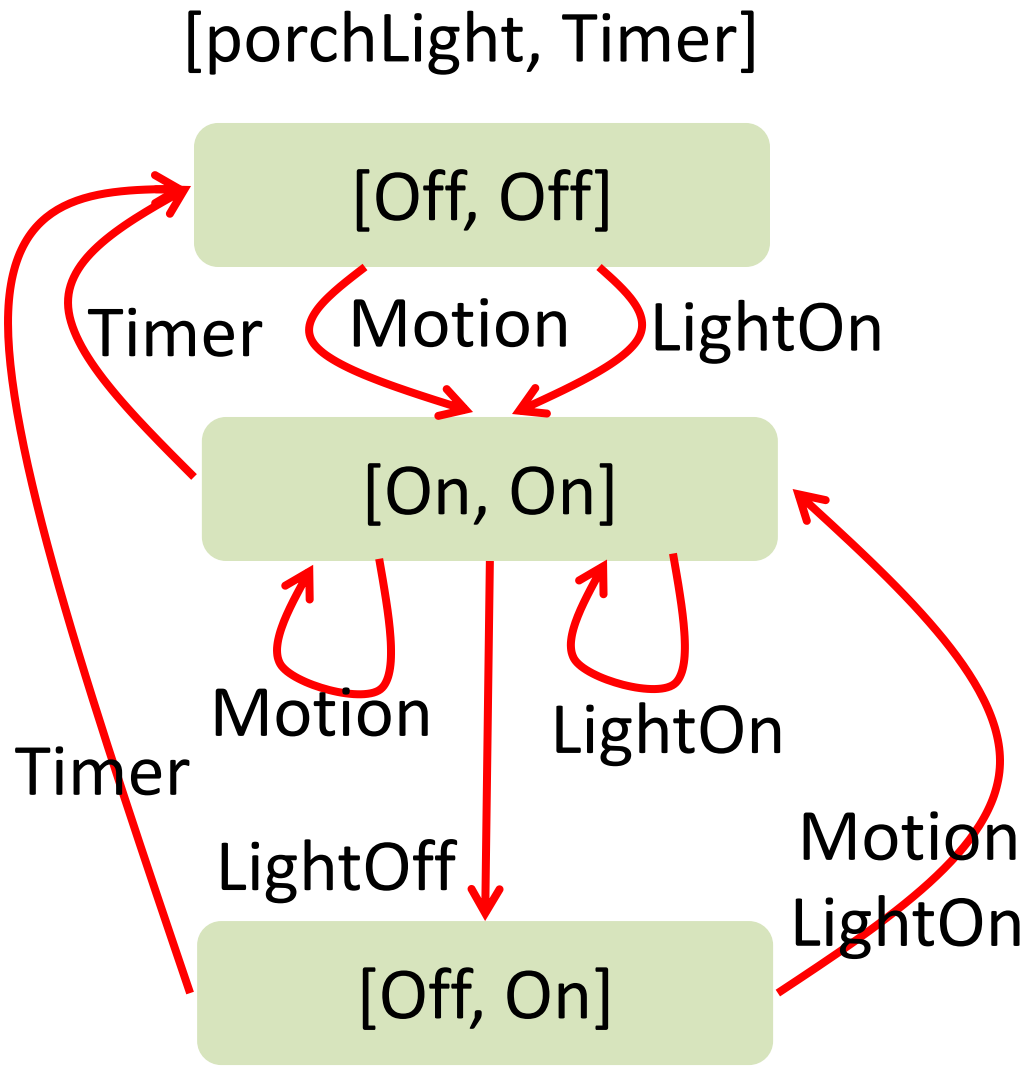
```
packetIn:
  entry = new Entry(inPkt.src,
                    inPkt.dst)
  if (!cache.Contains(entry))
      cache.Insert(entry, Now)

cleanupTimer:
  foreach entry in cache:
     if (Now - cache[entry] < 5s)
         cache.Remove(entry)
```

Unintended Behavior is a Concern

- Security and monetary risks
  - Unlocked doors, improperly activated thermostats, network/service outages
- We don’t want this happening at 8AM in the summer

Possible Solution: Model Check Code

```
motionPorch.Detected:
    porchLight.Set(On)
    timer.Start(1 min)

porchLight.On:
    timer.Start(1 min)

timer.Fired:
    porchLight.Set(Off)
```



Untimed Model Checking Isn't Enough

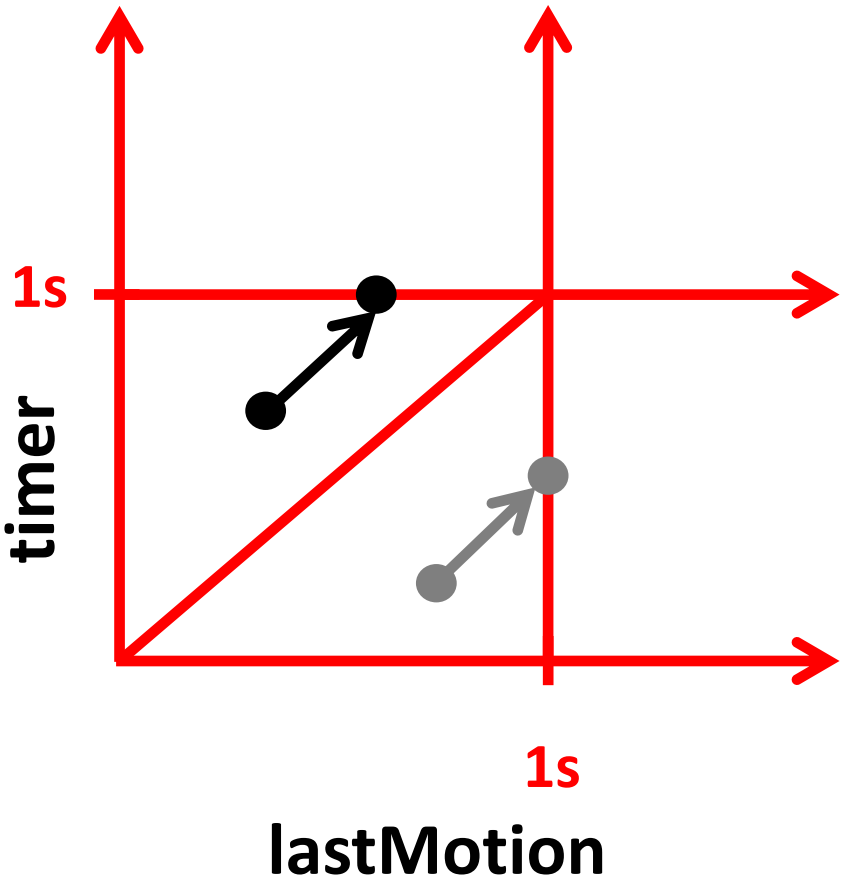
- **Problem:** anything can happen at any time
  - Event domain can be large
  - Time domain is continuous
  - How to reason about behavior under **all possible events at all possible times**?
  - How to check **time-bounded** properties?
- **Solution:**
  - All possible events => symbolic execution
  - All possible times => timed automata

Efficiently Exploring Temporal Behavior

- **Timed Automata (TA):** finite state machines (states , transitions) extended with real-valued *virtual clocks (VC)*
  - Divide time into equivalent *regions* using VC constraints
  - All VCs progress at same rate, except that one or more VCs may be reset on a transition
  - VC constraints gate transitions

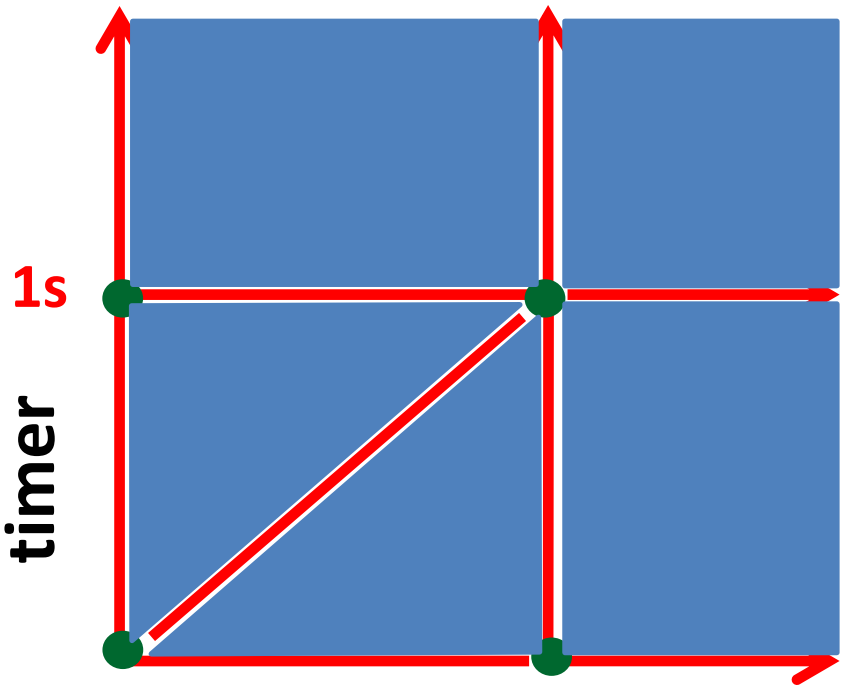
Timed Automata Regions: 2 VCs

```
motionPorch.Detected:
   if (Now – lastMotion < 1s)
        porchLight.Set(On)
        timer.Start(1s)
   lastMotion = Now

timer.Fired:
    porchLight.Set(Off)
```



Timed Automata Regions: 2 VCs

```
motionPorch.Detected:
   if (Now – lastMotion < 1s)
        porchLight.Set(On)
        timer.Start(1s)
   lastMotion = Now

timer.Fired:
    porchLight.Set(Off)
```



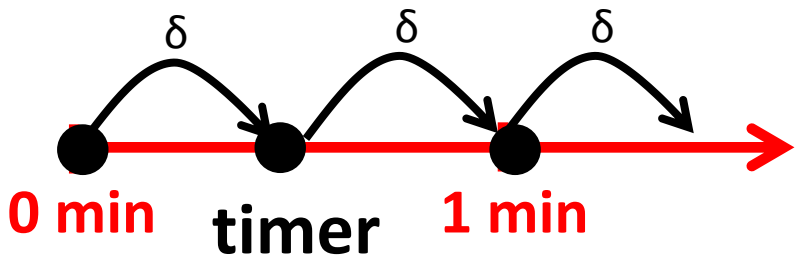
9 line segments

4 intersections

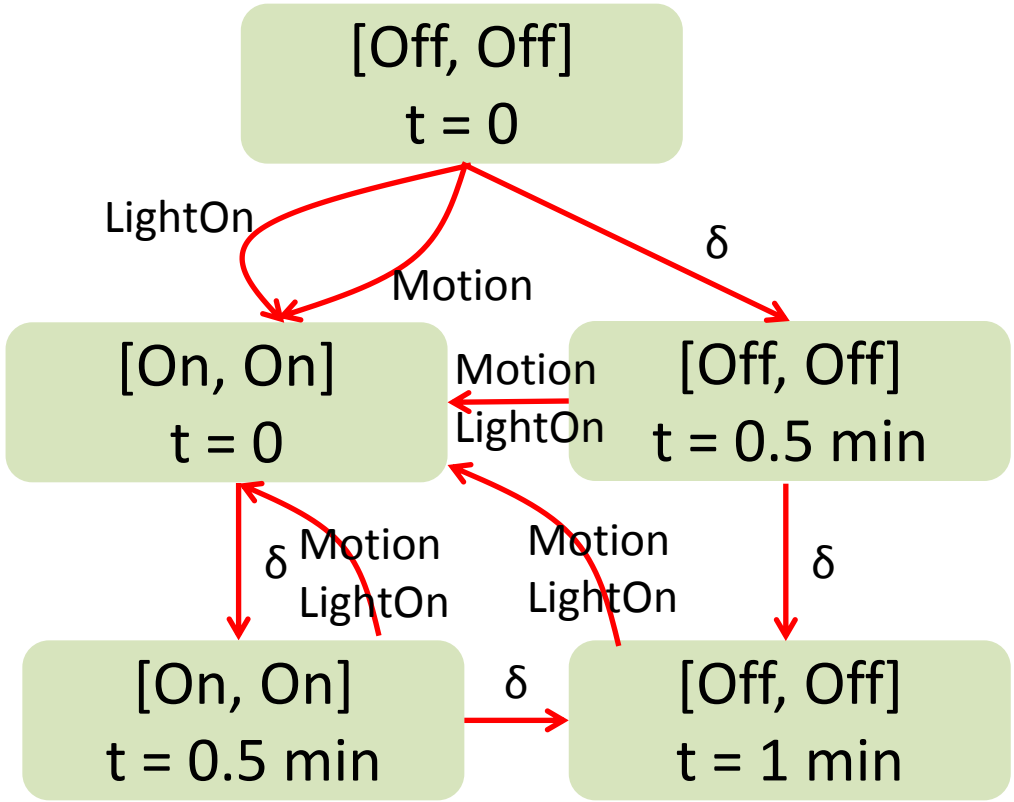
5 open spaces

Exploring a Timed Automaton

```
motionPorch.Detected:
    porchLight.Set(On)
    timer.Start(1 min)

porchLight.On:
    timer.Start(1 min)

timer.Fired:
    porchLight.Set(Off)
```

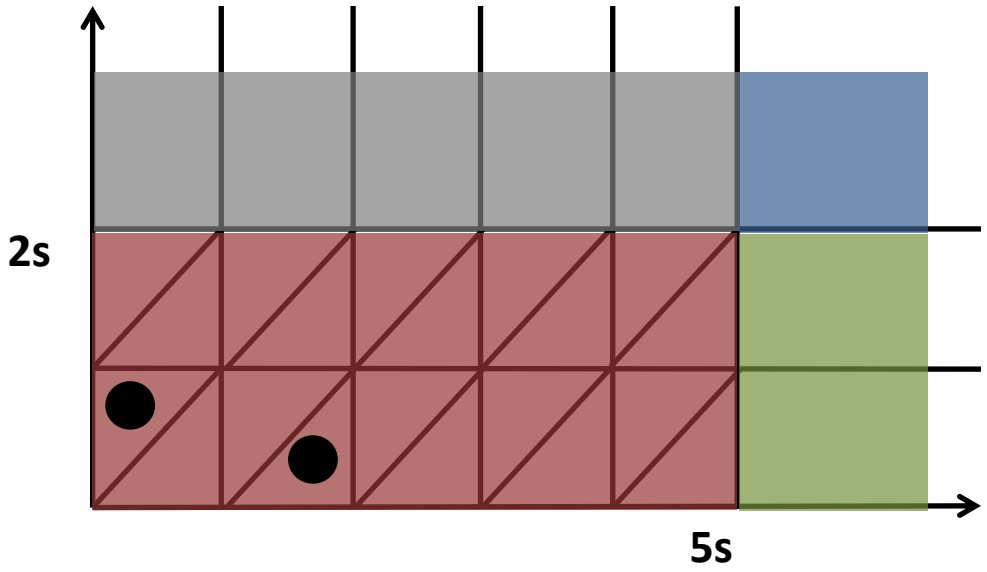


[porchLight, Timer]
t=timer



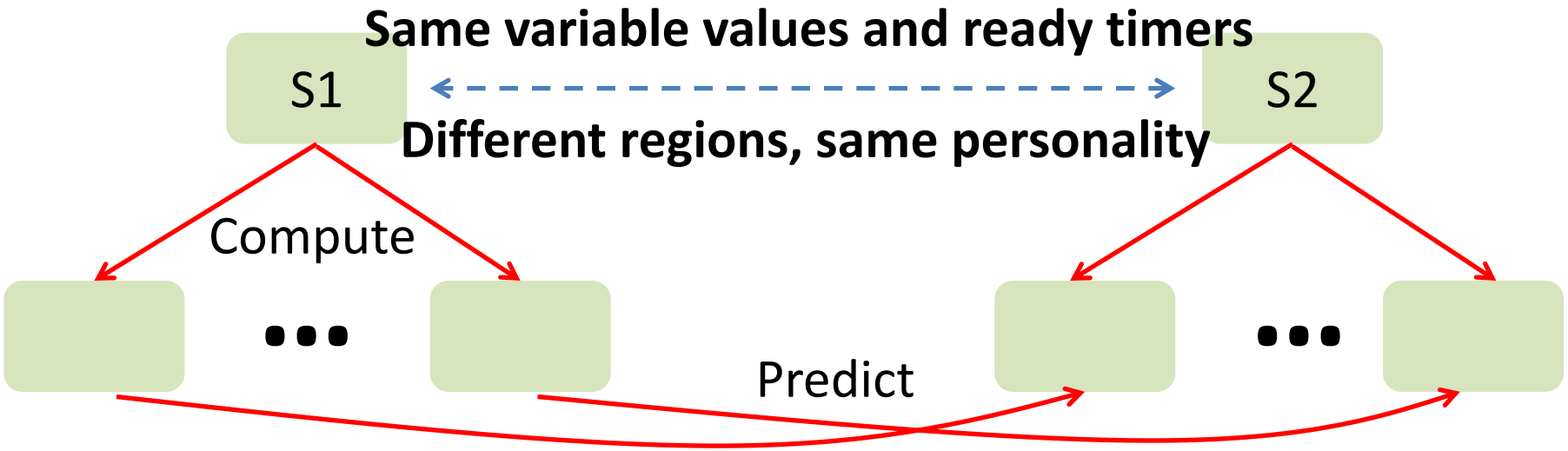
Optimization 1: Predicting Successors

- Problem: Running code is costly
- Observation: Multiple region states can have identical response to a trigger
- **Clock personality:** region's evaluation of a clock



Optimization 1: Predicting Successors

- Problem: Running code is costly
- Observation: Multiple region states can have identical response to a trigger
- **Clock personality:** region's evaluation of a clock



Optimization 2: Minimizing Virtual Clocks

- Problem: VCs impact efficiency
- Observation: VCs for some actions can be combined

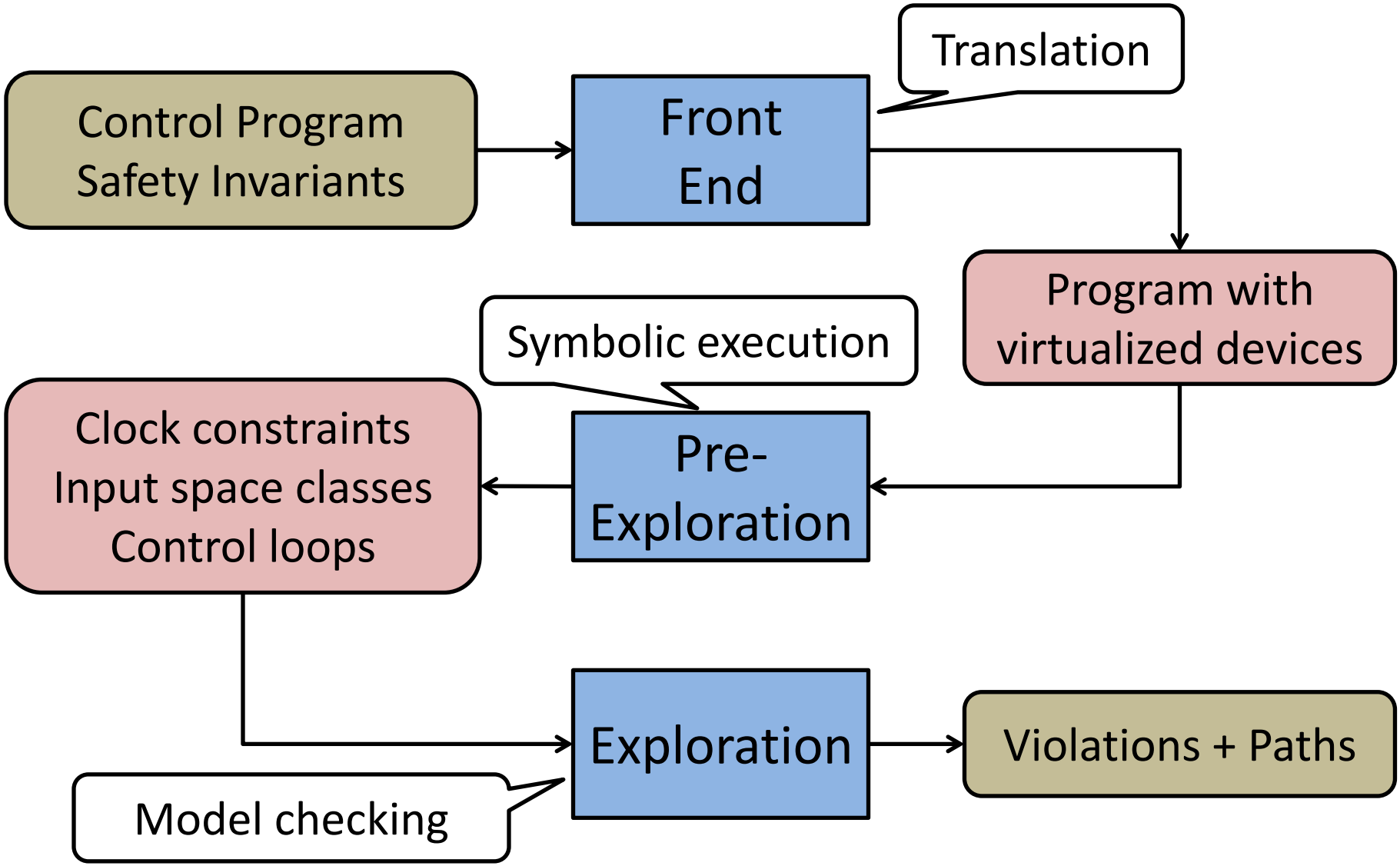
1. Multiple sleeps can be modeled with one VCs
   - Eg: `action1; sleep(5); action2; sleep(10);`
2. All time of day tasks can be modeled with one VC
   - Eg, actions at sunset, sunrise, 7PM

Optimization 3: Independent Loops

- Problem: more devices means more states
- Observation: control programs tend to have multiple, independent control loops

1. Determine independent sets of variables
2. Explore each set independently

DeLorean: Workflow



Testing Home Automation Scripts

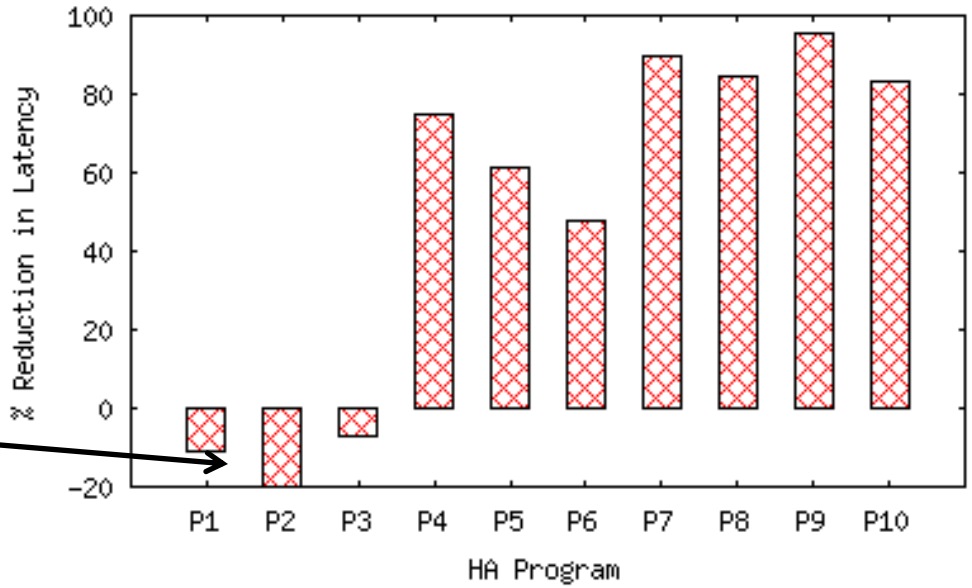
- 10 real home automation programs
  - Between 3 and 51 devices
  - Between 3 and 90 rules (triggers)
  - Between 2 and 14 virtual clocks

- Find 4 bugs
  - 3 of which are time-related

Testing Home Automation: Performance

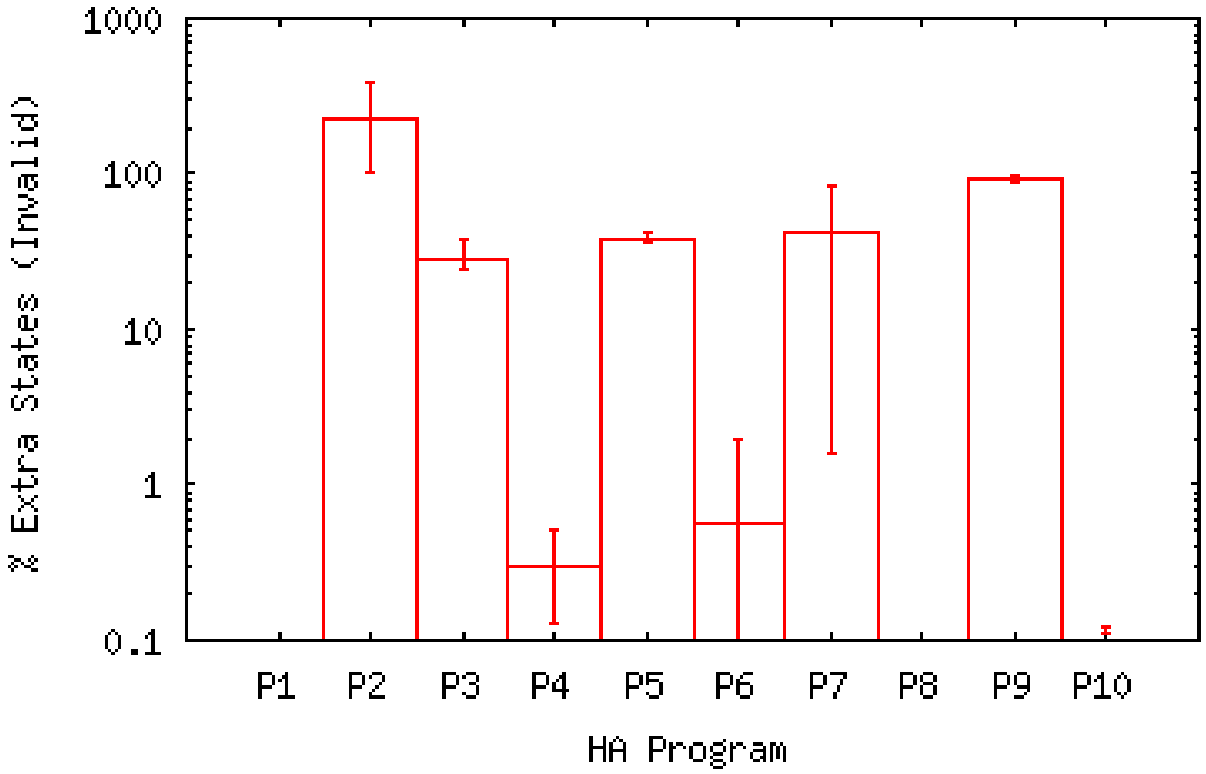
- DeLorean explores programs 3.6 to 36K times faster than real time
  - ~200k states/sec
- Predicting states: up to 90% reduction in exploration time

No prediction, overhead in checking for similar states



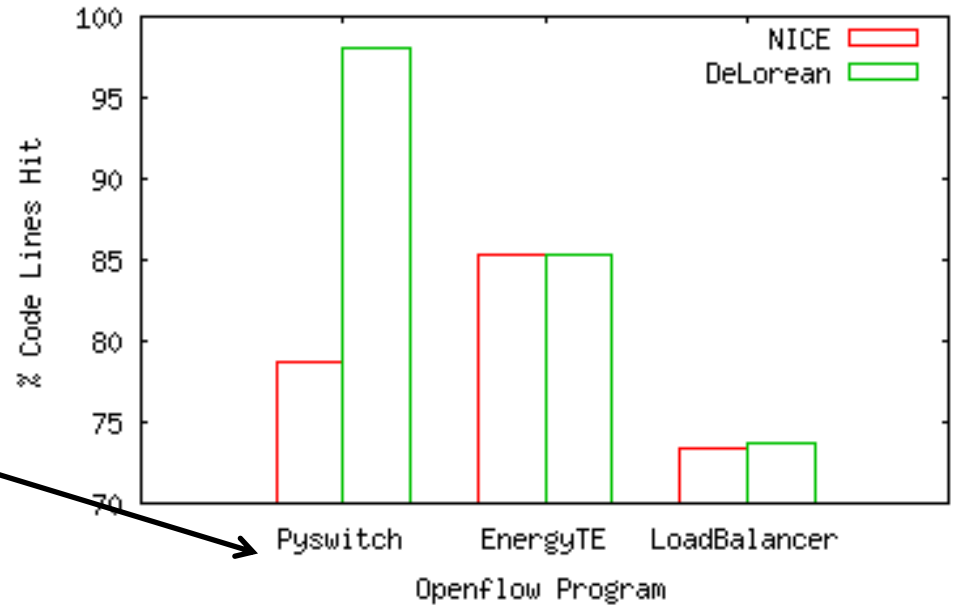
Testing Home Automation: Correctness

- Compared to untimed model checking



Testing SDN Apps

- 3 SDN apps:
  - MAC-learning switch, traffic engineering, load balancer
- Compare with NICE – untimed model checker



Largest dependency on time

Conclusion + Lessons Learned

- Can't forget about time when testing
  - Even simple programs can have complex dependencies on time

- TA-based exploration most useful for programs with dependence on absolute (time-of-day) *and* relative time