GLOBAL INITIATIVE OF ACADEMIC NETWORKS

University of Illinois at Urbana-Champaign

Modeling and verification of cyber-physical systems

Lecture 1

Jan 1-5st 2018

Sayan Mitra[1] and Purandar Bhaduri[2]

[1]University of Illinois at Urbana-Champaign

[2]Indian Institute of Technology Guwahati

mitras@Illinois.edu

Plan for the lecture 1

- Course administration
- Motivation
  - What are cyber-physical systems?
  - Why care about verification of CPS?
- Modeling Discrete Systems
- Invariance and safety

Algorithms make decisions for us everyday

Lecture Slides by Sayan Mitra
mitras@illinois.edu

Increasingly, algorithms control our physical environment and social institutions

Lecture Slides by Sayan Mitra
mitras@illinois.edu

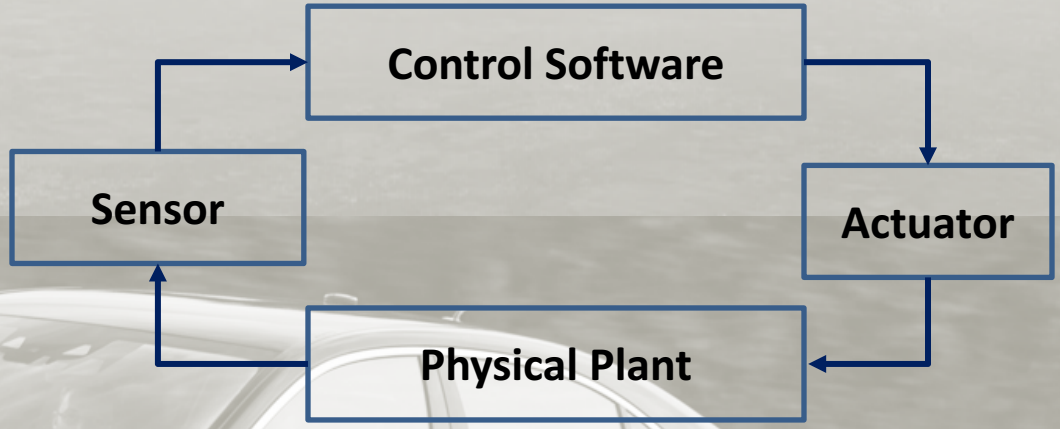
Lecture Slides by Sayan Mitra
mitras@illinois.edu

Cyber-physical Systems

CPS: software monitoring and controlling a dynamical system

We will take a rigorous view of modeling, analyzing, and designing engineering systems

Why?



Lecture Slides by Sayan Mitra
mitras@illinois.edu

Autonomous systems

Internet Of Things (IOT)

Cyber-physical System

Industry 4.0

Lecture Slides by Sayan Mitra
mitras@illinois.edu

Is the algorithm fair?

Advanced Driving Assist features (ADAS) to Driverless cars

Lecture Slides by Sayan Mitra
mitras@illinois.edu

2004 DARPA Grand challenge I:

no car drives more than 7 miles in desert

2005 DARPA challenge II:

Stanford car wins driving 131 miles

2007 DARPA Urban challenge*

Many successes, CMU wins

Google, Uber, Tesla, GM, Nissan, Toyota, Ford, …

2016: Public tests at Singapore (nuTonomy), Pittsburgh (Uber), Boston

Verification of Periodic Hybrid Systems: Application to An Autonomous Vehicle
Wongpiromsarn, Mitra, Lamperski and Murray 2009

Algorithms for predictive policing

Lecture Slides by Sayan Mitra
mitras@illinois.edu

Crime records + Surveillance -> Predictions

2008: LAPD starts explorations on forecasting crime using data
2013: Better prediction of crime hotspots in Santa Cruz evaluation
2016: Used in 50+ police department

Zach Friend. "Predictive Policing: Using Technology to Reduce Crime". Federal Bureau of Investigation. Dec. 2013.

Lecture Slides by Sayan Mitra
mitras@illinois.edu

Technology is neither good nor bad; *nor is it neutral*.

Lecture Slides by Sayan Mitra
mitras@illinois.edu

Driverless cars will be good for

Productivity: Americans spend 300 hours driving each year

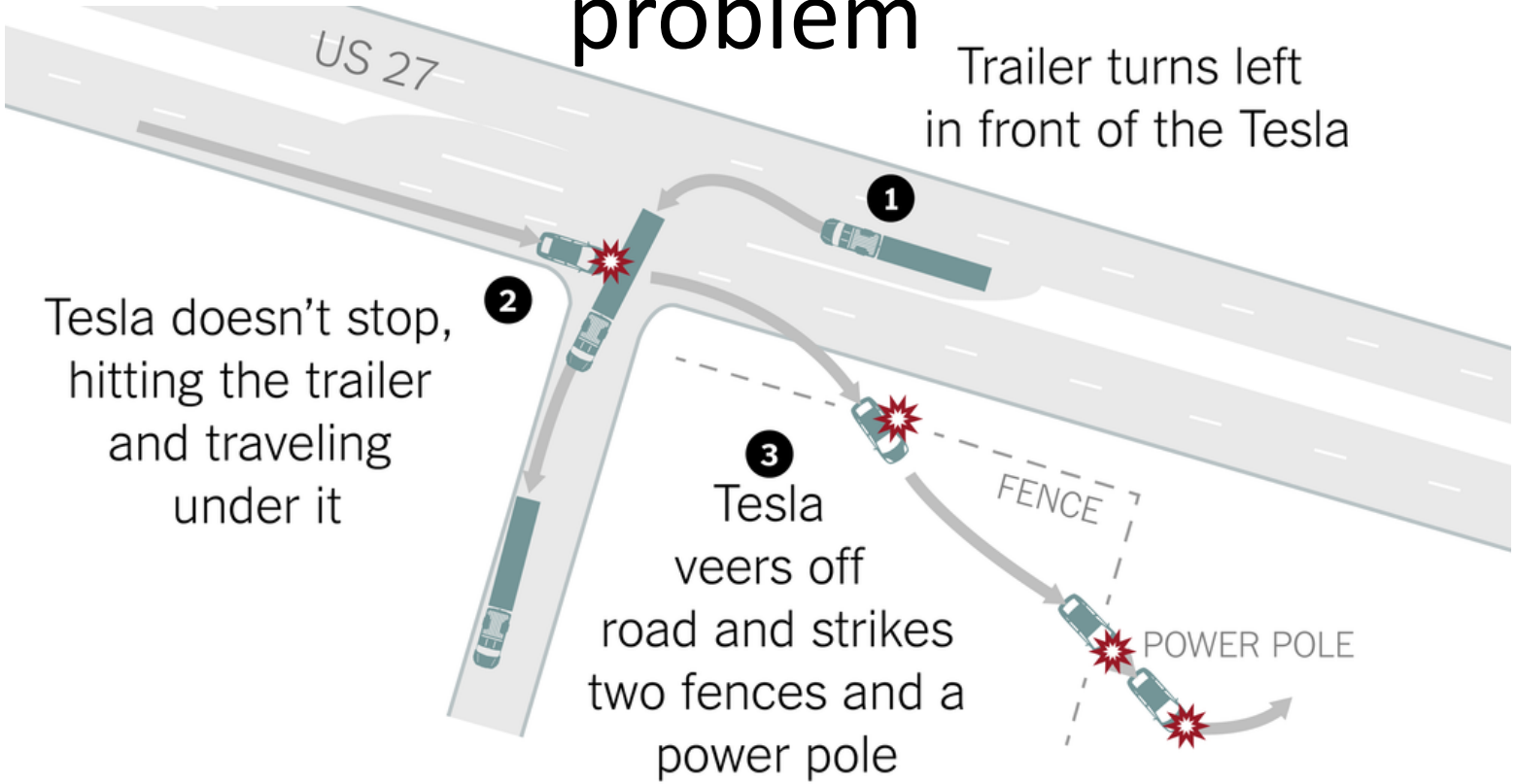
Real estate: 40 % of city surface is parking

Environment: Better fuel efficiency, scheduling, charging

Safety: 32,675 vehicle fatalities in 2014 in the US

10% increase in first half of 2016 !!

Achieving safety is a very hard problem



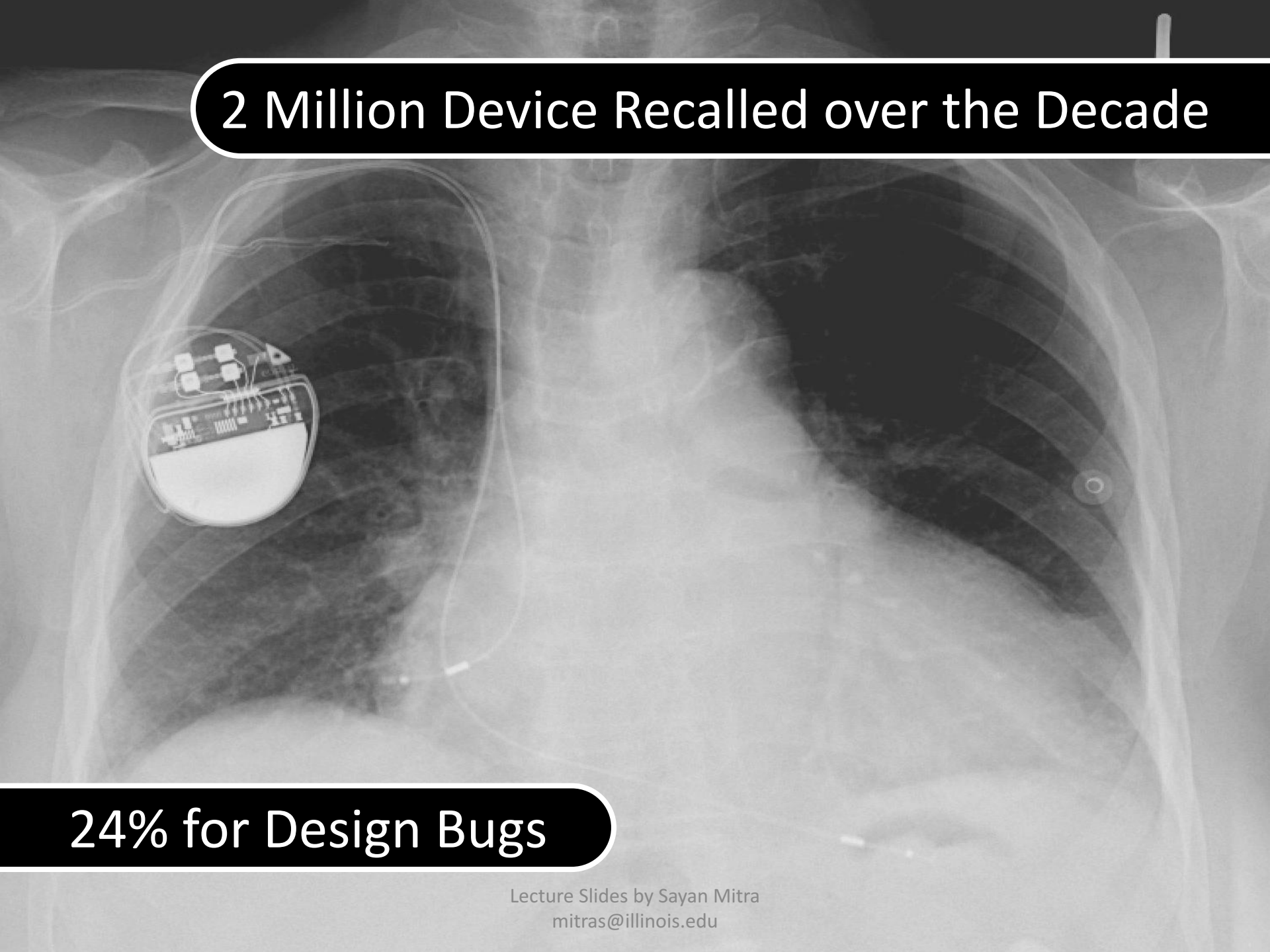
NYTimes, July 12 2016

A Record 22 Million Cars Recalled in 2013

Cost in Billions

Lecture Slides by Sayan Mitra
mitras@illinois.edu

2 Million Device Recalled over the Decade



24% for Design Bugs

Lecture Slides by Sayan Mitra
mitras@illinois.edu

ZA002 Incident

EVERETT, Wash., Nov. 10, 2010-- During approach to Laredo, Texas, yesterday, airplane ZA002 lost primary electrical power as a result of an onboard electrical fire. Backup systems, including the deployment of the Ram Air Turbine (RAT), functioned as expected and allowed the crew to complete a safe landing. The cause of the fire is still under investigation by Boeing.

Initial inspection appears to indicate that a power control panel in the aft electronics bay will need to be replaced on ZA002. We are inspecting the power panel and surrounding area near that panel to determine if other repairs will be necessary.

Boeing: Changes to Power Panel Design Nov. 24, 2010
Boeing has acknowledged the rumors circulating around that it was indeed FOD (foreign object damage) that had caused a short circuit that led to the fire in the P100 power panel.

As a consequence Boeing will undertake a minor redesign of the power panel to reduce the chance of FOD creating and electrical arc or short circuit. Boeing will also implement software changes to make sure that power distribution is improved. This appears to be an acknowledgement that the electrical power redundancies failed as well.

New in DO178C standard

- Formal Methods
- Model Based Development
- Object Oriented Techniques

Lecture Slides by Sayan Mitra
mitras@illinois.edu

Overflow results in the flight computer crash (2005)

"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

- 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 the program and its physical environment

How many miles must an autonomous car drive before we call it safe?

200 million miles?

0.07 fatalities per billion passenger miles (commercial flight)

Probability of fatal failure per hour of driving 10^{-9} [Amnon Shashua, CTO Mobileye]

How is air-travel so safe?

Lecture Slides by Sayan Mitra
mitras@illinois.edu

Regulations and Audits

What fraction of the cost of developing a new aircraft is in SW?

DO178C

Primary document by which FAA & EASA approves software-based aerospace systems.

DAL establishes the rigor necessary to demonstrate compliance

Dev.Assurance Level (DAL)	Hazard Classification	Objectives
A	Catastrophic	71
B	Hazardous	69
C	Major	62
D	Minor	26
E	No Effect	0

Statement Coverage: Every statement of the source code must be covered by a

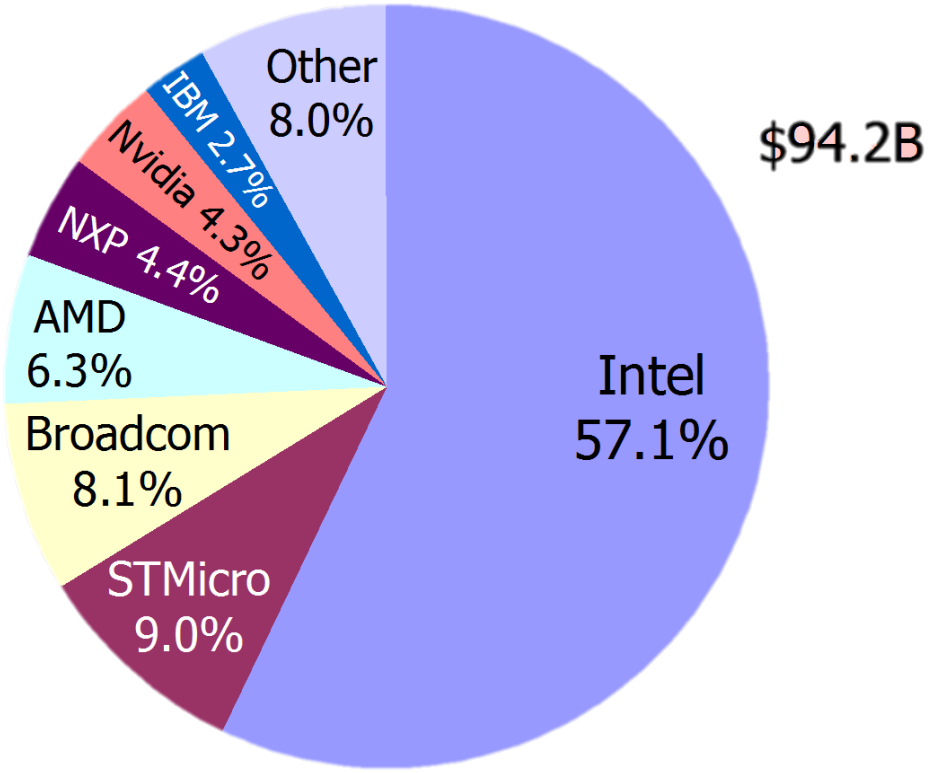
Condition Coverage: Every condition within a branch statement must be covered by a test case

Lecture Slides by Sayan Mitra
mitras@illinois.edu

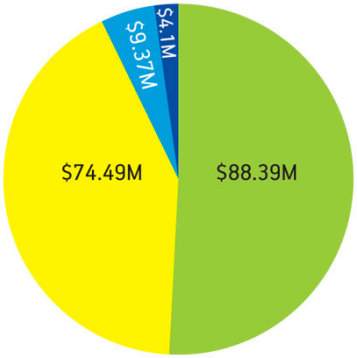
Verification of CPS:
We need standards, processes, tools, and trained individuals to ensure that cyber-physical systems meet the standards

Lecture Slides by Sayan Mitra
mitras@illinois.edu

An earlier instance: microprocessor industry



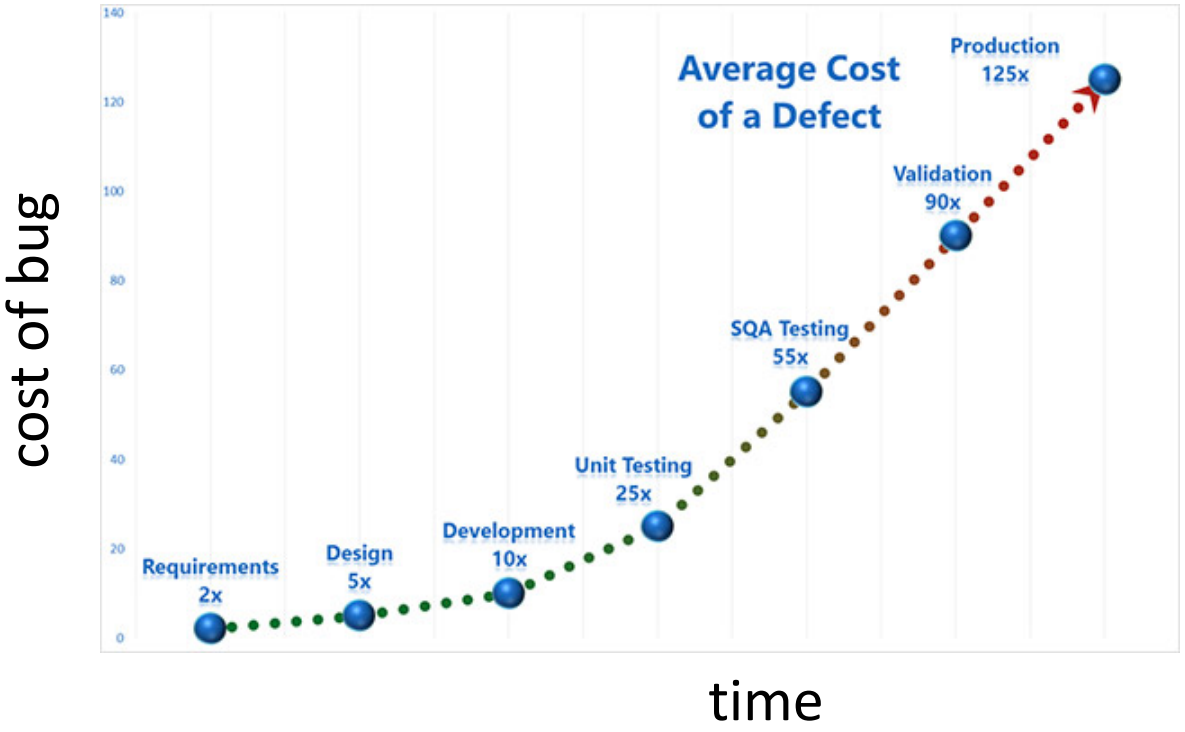
$94.2B



Electronic design automation industry

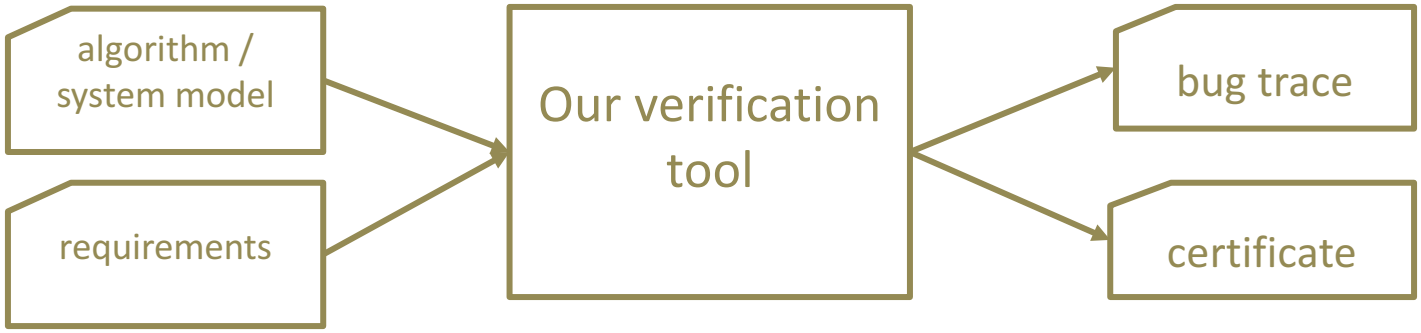
- Synopsys 51%
- Mentor Graphics 42%
- Cadence Design Systems 5%
- Other 2%

Defects become more expensive with time



How to Cut Software-Related Medical Device Failures and Recalls, Lisa Weeks

Audit algorithms with Algorithms and find problems early



Relevant courses: Theory of computation, Program Verification, Formal System Development, Automated Deduction, Control theory, Embedded System Verification

Lecture Slides by Sayan Mitra
mitras@illinois.edu

Example requirements

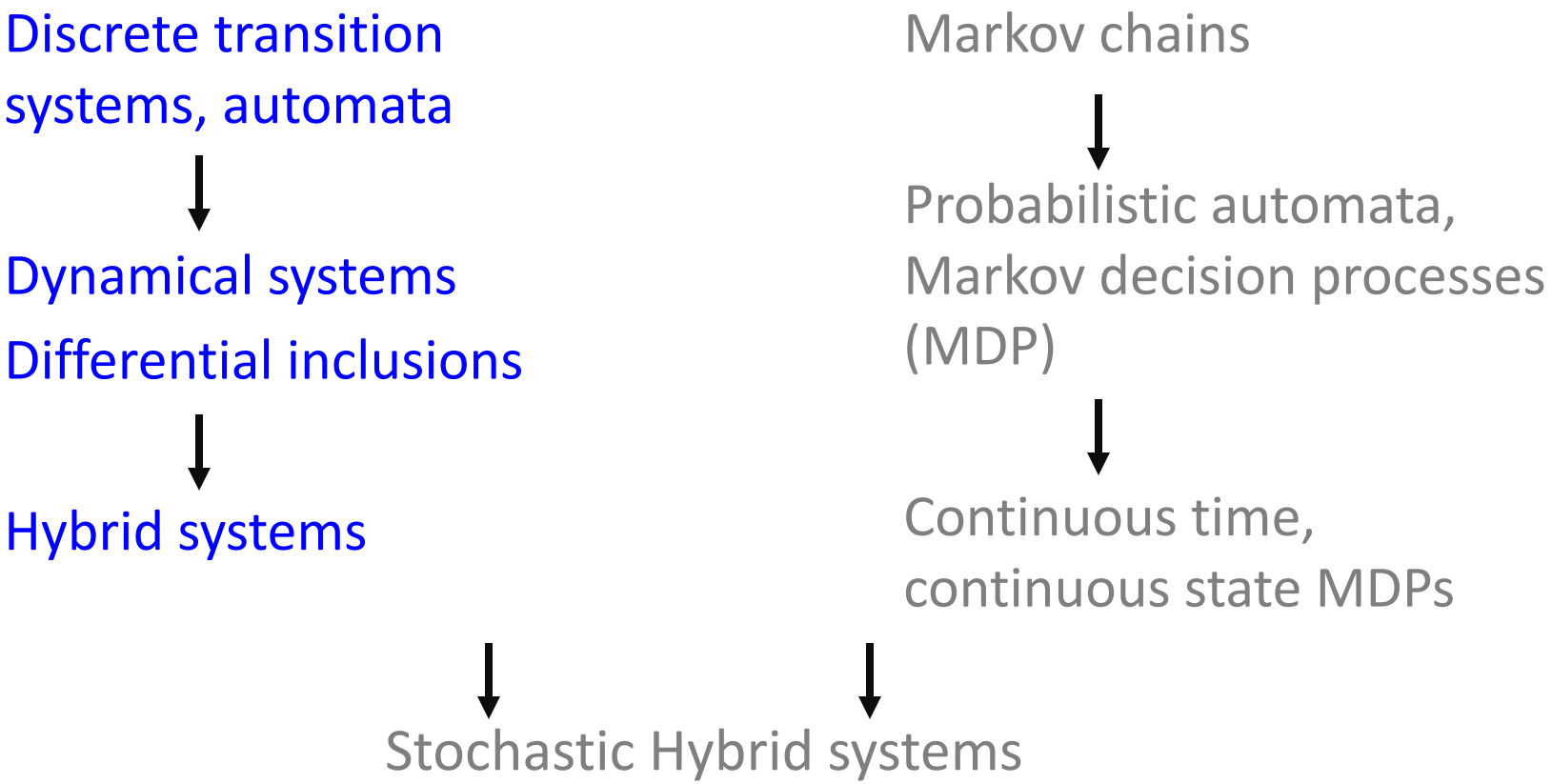
Safety: “For all nominal behaviors of the car, the separation between the cars must be always > 1 m”

Efficiency: “For all nominal driver inputs, the air-fuel ratio must be in the range [1,4]”

Privacy: “Using GPS does not compromise user’s location”

Fairness: “Similar people are treated similarly”

Example modeling frameworks



Young & promising area with rich problems

or why you should stop worrying and love this course

- Intersection of CS and control theory
  - Discrete and continuous math, different analysis techniques
- Formal analysis gaining traction in industry
  - Hardware verification is engineering practice in the industry
  - Many tools used in software, avionics, automotive industries. e.g. SDV (Microsoft), ESTEREL, ASTREE.
  - Mars Code: Gerard Holzman, JPL (see video)
- Big and small commercial enterprises support above customers
  - Synopsis, Mentor Graphics, Cadence, Jasper, Coverity, Galois, SRI,
- Vibrant academic research activities
  - Related Turing Awards: Lamport (2014), Clarke, Sifakis & Emerson (2008), Pnueli (1997), Lampson (1992), Milner (1991), Hoare (1980), Dijkstra (1972) …
  - Conferences: HSCC, EMSoft, ICCPS, CAV, TACAS, RTSS…
  - Alums from this course now faculty at U. Conn, UT Arlington, U. Minnesota, Kansas State, Air Force Research Lab, Toyota Research Center, …

Learning Objectives

- Techniques for **modeling** systems with dynamics, computation, and communication
- Techniques for **rigorously reasoning** about correctness of systems
- Exposure to using **verification tools** and libraries
  - model checkers, SMT solvers, simulation-driven verifiers ~~and theorem provers~~
- **Understand** deep and influential ideas in CS and Control theory (from the last 20 years)

GLOBAL INITIATIVE OF ACADEMIC NETWORKS

Indian Institute of Technology Guwahati

University of Illinois at Urbana-Champaign

Modeling Computation

DISCRETE SYSTEMS

Lecture Slides by Sayan Mitra
mitras@illinois.edu

Outline

- An Example: Token Ring
- Specification language (syntax)
- Automata (semantics)
- Invariants

An example: Informal description

A **token-based** mutual exclusion algorithm on a **ring network**

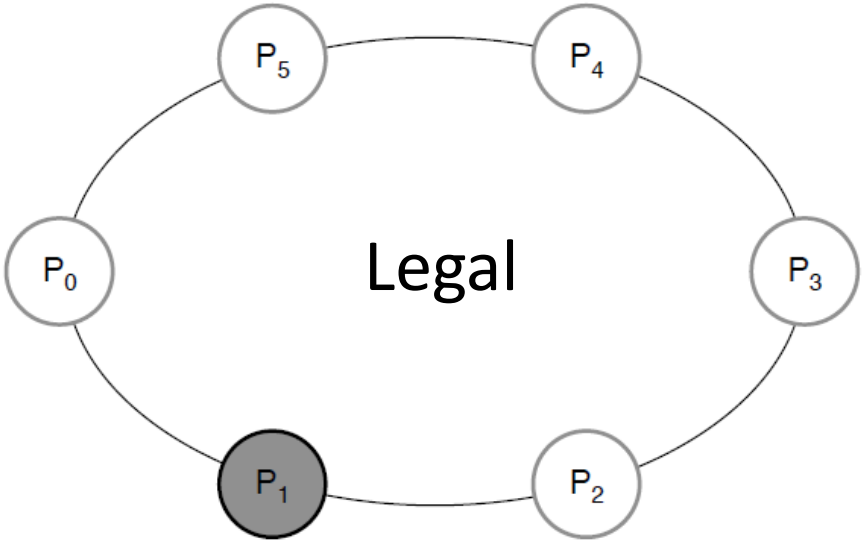
Collection of processes send and receive bits over a ring network so that only one of them has a “token”

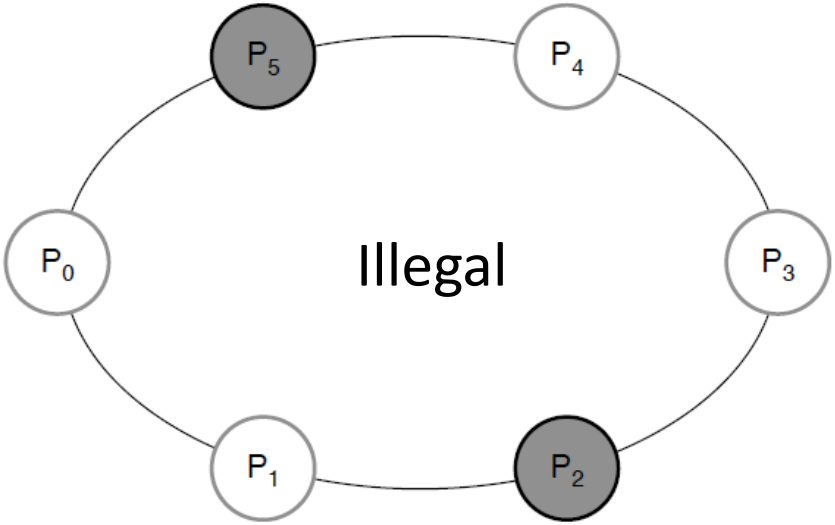
Discrete

Each process has variables that take only **discrete values**

Time elapses in **discrete steps** (This is a modeling choice)

Token ring: Informal problem specification



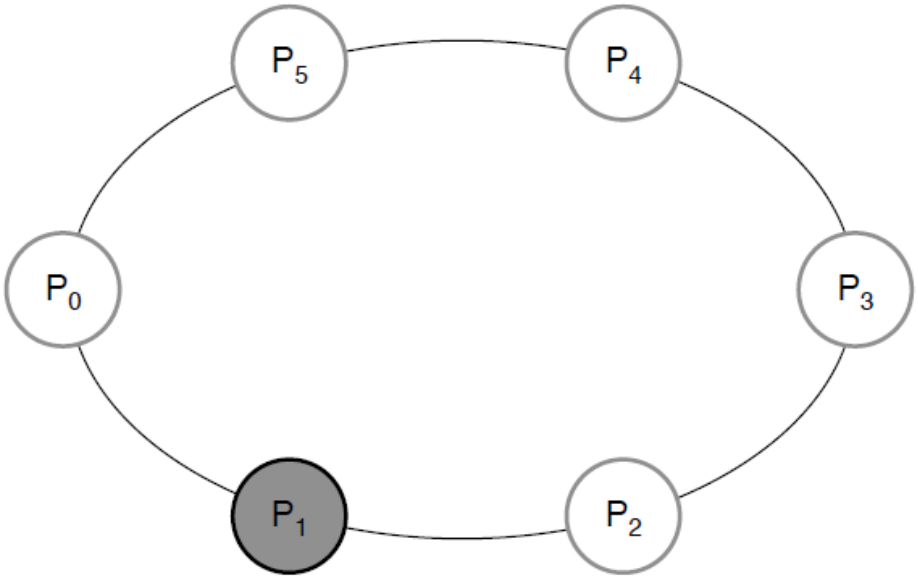


1. There is always at least one token
2. Legal configuration = exactly one “token” in the ring
3. Single token circulates in the ring
4. Even if multiple tokens somehow arise, e.g. with failures, if the algorithm continues to work correctly, then eventually there is a single token

Properties can be stated as Invariants

- **Invariant** (informal def.): A property of the system that always* holds
- Examples:
  - "Always at least one process has a token"
  - "Always exactly one process has the token"
  - "Always all processes have values at most k-1"
  - "Even if there are multiple tokens, **eventually** there is exactly one token" (not strictly an invariant)

Dijkstra's Algorithm [Dijkstra 1982]



n processes with indices 0, 1, …, n-1

state of process j is x[j] $\in$ {0, 1, 2, k-1}, where k > n

p_0 **if** x[0] = x[N-1] **then** x[0] := x[0] + 1 mod k

p_j j > 0, **if** x[j] ≠ x[j -1] **then** x[j] := x[j-1]

(p_i **has TOKEN** if and only if the conditional is true)

Lecture Slides by Sayan Mitra
mitras@illinois.edu

A Specification Language: HIOA

auto DijkstraTR (n:natural,k:natural)
type indices: [0,…,n-1]
type values: [0,…,k-1]
signature
    internal step(i:indices)
variables
    x:[indices->values] **initially** $\forall i \in$ indices, $x[i] = 0$
transitions
    internal step(i:indices)
    pre i = 0 /\ x[i] = x[n-1]
    eff x[i] := x[i] + 1 mod k;

    internal step(i:indices)
    pre i ≠ 0 /\ x[i] ≠x[i-1]
    eff x[i] := x[i-1];

trajectories

Discrete Transition System or Automaton

An **automaton** is a tuple $\mathcal{A} = \langle X, \Theta, A, \mathcal{D} \rangle$ where

1. X is a set of names of variables; each variable $x \in X$ is associated with a type, $type(x)$
   - A **valuation** for X maps each variable in X to its type
   - Set of all valuations:$val(X) = Q$ this is sometimes identified as the **state space** of the automaton
2. $\Theta \subseteq val(X)$ is the set of **initial** or **start states**
3. A is a set of names of **actions** or **labels**
4. $\mathcal{D} \subseteq val(X) \times A \times val(X)$ is the set of **transitions**
   - a transition is a triple (u, a, u')
   - We write $(u, a, u') \in \mathcal{D}$ in short as $u \xrightarrow{a} u'$

HIOA Specs to Automata: variables

variables s, v: Reals; a: Bools

$X = \{s, v, a\}$

Example valuations also called **states**:

- $u_1 = \langle s \mapsto 0, v \mapsto 5.5, a \mapsto 0 \rangle$
- $u_2 = \langle s \mapsto 10, v \mapsto -2.5, a \mapsto 1 \rangle$

$val(X) = \{\langle s \mapsto c_1, v \mapsto c_2, a \mapsto c_3 \rangle | \; c_1, c_2 \in R, c_3 \in \{0,1\}\}$

type indices: [0,…,n-1]

variables x: [indices->values]

- Fix n = 6, k = 8
- x: [{0,…,5} -> {0,…,7}]
- Example valuations:
  - $u = \langle x \mapsto \langle 0 \mapsto 0, 1 \mapsto 0, 2 \mapsto 0, 3 \mapsto 0, 4 \mapsto 0, 5 \mapsto 0 \rangle \rangle$
  - $v = \langle x \mapsto \langle 0 \mapsto 7, 1 \mapsto 0, 2 \mapsto 0, 3 \mapsto 0, 4 \mapsto 0, 5 \mapsto 0 \rangle \rangle$
  - Notation: $\boldsymbol{u}.x, \boldsymbol{u}.x[4]$ =0

$val(x) = \left\{ \langle x \mapsto \langle i \mapsto c_i \rangle_{\{i=0\ldots5\}} \rangle \middle| c_i \in \{0, \ldots, 7\} \right\}$

States and predicates

A **predicate** over a set of variables X is a formula involving the variables in X. For example:

- $\phi_1 : \mathrm{x}[1] = 0$
- $\phi_2 : \forall i \in indices, x[i] = 0$

A valuation **u satisfies predicate ϕ** if substituting the values of the variables in **u** in ϕ makes it evaluate to **True.** We write **u$\models \phi$**

- $\boldsymbol{u} \models \phi_1, \boldsymbol{u} \models \phi_2, \boldsymbol{v} \models \boldsymbol{\phi_1}$ and $\boldsymbol{v} \not\models \boldsymbol{\phi_2}$

$[[\boldsymbol{\phi}]] = \{u \in val(x) | \, u \models \phi\}$. Examples

- $[[\phi_1]] = \{\langle x \mapsto \langle 1 \mapsto 0, i \mapsto c_i \rangle_{\{i=0,2,\ldots,5\}} \rangle | c_i \in \{0, \ldots, 7\}\}$
- $[[\phi_2]] = \{\langle x \mapsto \langle 0 \mapsto 0, 1 \mapsto 0, 2 \mapsto 0, 3 \mapsto 0, 4 \mapsto 0, 5 \mapsto 0 \rangle\rangle\}$

Initial state and invariant assertions

- $\Theta \subseteq val(x)$ initial states
  - Often specified by a predicate
  - $\boldsymbol{\phi_0} = (\textbf{Initially } \forall \boldsymbol{i} \in \text{indices}, x[i] = 0)$
  - $\Theta = [[\phi_0]] = \langle x \mapsto \langle i \mapsto 0 \rangle_{i=0,\ldots,5} \rangle$

- Invariant properties
  - "At least one process has the token".
  - $\boldsymbol{I_1} = (x[0] = x[5] \vee \exists\, i \in \{1, \ldots 5\}: x[i] \neq x[i-1])$
  - $[[I_1]] = \{\langle 0, \ldots, 0\rangle, \langle 1,0, \ldots, 0\rangle, \ldots, \langle k-1, \ldots, k-1\rangle\} = val(x)$ *(?)*
  - "Exactly one process has the token"
  - $\boldsymbol{I_2} = (x[0] = x[5] \oplus x[1] \neq x[0] \oplus x[2] \neq x[1] \ldots)$

Actions

- **signature** defines the set of Actions
- Examples
  - **internal** step(i:indices)
  - $A = \{step[0], \ldots, step[5]\}$
  - **internal** brakeOn, brakeOff
  - $A = \{brakeOn, brakeOff\}$

Transitions

$\mathcal{D} \subseteq val(X) \times A \times val(X)$ is the set of **transitions**

internal step(i:indices)
  pre i = 0 /\ x[i] = x[n-1]
  eff x[i] := x[i] + 1 mod k;

internal step(i:indices)
  pre i ≠ 0 /\ x[i] ≠x[i-1]
  eff x[i] := x[i-1];

$(u, a, u') \in \mathcal{D}$ iff $u \vDash Pre_a$ and $(u, u') \in Eff_a$

$(u, step(i), u') \in \mathcal{D}iff$

(a) $(i = 0 \wedge u.x[0] = u.x[5]$

$\wedge u'.x[0] = u.x[0] + 1 \ mod \ 6)$ V

(b)$(i \neq 0 \wedge u.x[i] \neq u.x[i-1]$

$\wedge u'.x[i] = u.x[i-1])$

Nondeterminism

- For an action $a \in A$, Pre(a) is the formula defining its **pre**condition, and Eff(a) is the relation defining the **eff**ect.
- States satisfying precondition are said to **enable** the action
- In general Eff(a) could be a relation, but for this example it is a function
- **Nondeterminism**
  - Multiple actions may be enabled from the same state
  - There may be multiple post-states from the same action

Executions, Reachability, & Invariants

An **execution** of $\mathcal{A}$ is an alternating (possibly infinite) sequence of states and actions

$\alpha = u_0 a_1 u_1 a_2 u_3$ …such that:

- $u_0 \in \Theta$
- $\forall\, i$ in the sequence, $u_i \xrightarrow{a_{i+1}} u_{i+1}$

A state u is **reachable** if there exists an execution that ends at u. The set of reachable states is denoted by $Reach_A$.

Invariants (Formal)

What does it mean for I to hold "always" for $\mathcal{A}$?

- I holds at all states along any execution $u_0 a_1 u_1 a_2 u_3$
- I holds in all reachable states of $\mathcal{A}$
- $\boldsymbol{Reach_{\mathcal{A}}} \subseteq [[\boldsymbol{I}]]$

Invariants capture most properties that you will encounter in practice

- safety: "aircraft **always** maintain separation"
- bounded reaction time: "within 15 seconds of press, light must turn to walk"

How to **verify** if I is an invariant?

- Does there exist reachable state u such that $u \not\models I$?

Reachability Problem

- Given a directed graph $G = (V, E)$, and two sets of vertices $S, T \subseteq V$, T **is reachable from** S if there is a path from S to T.

- Reachability Problem (G, S, T) : decide if T is reachable from S in G.

Algorithm for deciding Reachability G,S, T

```
Set Marked := {}
Queue Q := S
Marked := Marked ∪ S
while Q is not empty
        t ← Q.dequeue()
        if t ∈ T return "yes"
        for each (t,u) ∈ E
                if u ∉ Marked then
                Marked := Marked ∪ {u}
                Q := enqueue(Q, u)
return "no"
```

Verifying Invariants by solving Reachability

Given $\mathcal{A} = \langle X, \Theta, A, \mathcal{D} \rangle$ and a candidate invariant I, how to check that I is indeed an invariant of $\mathcal{A}$?

Define a graph $\mathrm{G} = \langle V, E \rangle$ where

$$V = val(X)$$

$$E = \{(u, u') | \exists\, a \in A, u \xrightarrow{a} u'\}$$

Claim. $\left[\left[I\right]\right]^c$ is not reachable from Θ in G iff I is an invariant of $\mathcal{A}$.

Summary

- Well-formed specifications in the HIOA language define automata
- **Invariants**: Properties that hold at all reachable states. $Reach_{\mathcal{A}} \subseteq [[I]]$
- BFS to **verify invariants** automatically for (finite) automata