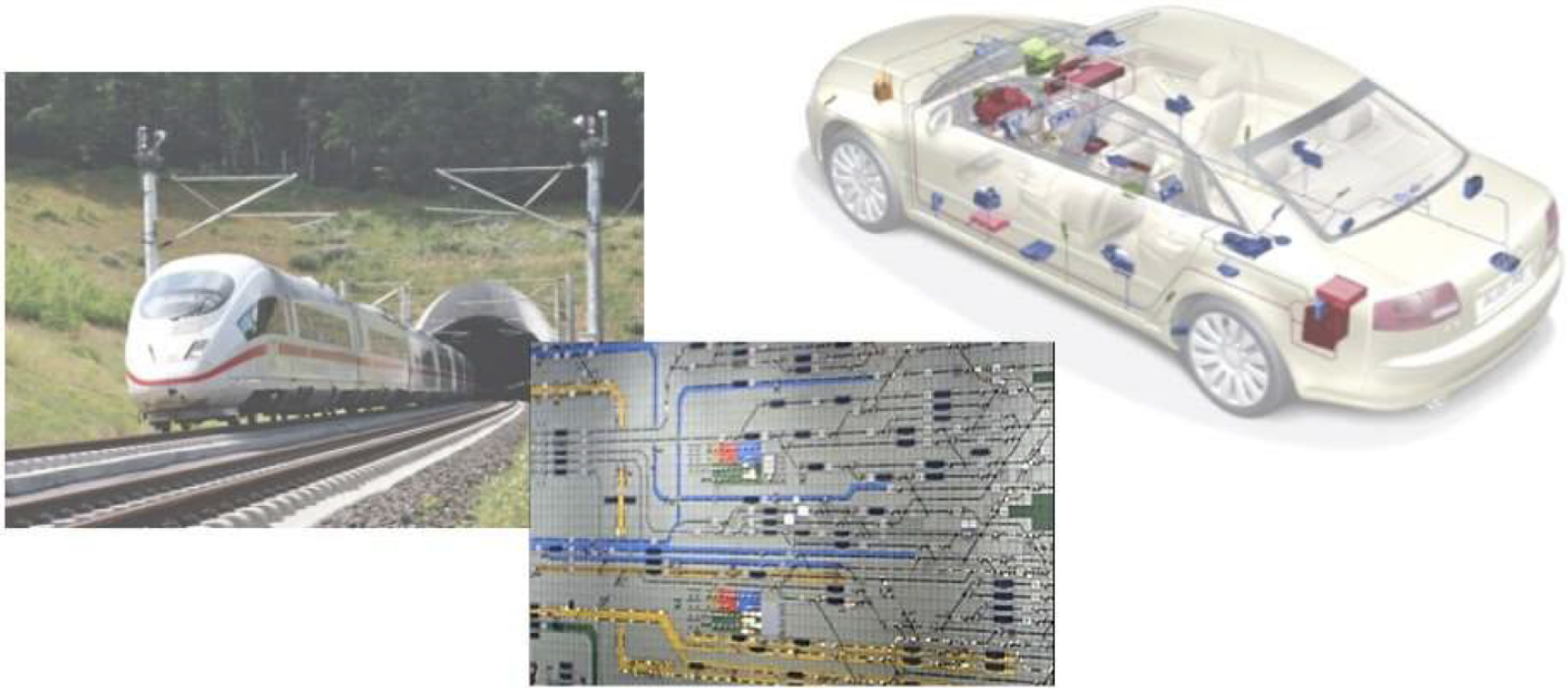
Model Checking and Systems Verification (MCSV)

Instructor; M.K. Srivas
TA: Zubin Duggal

August 14, 2020

Embedded Systems



- Embedded (HW+SW+FW) systems are a complex parts of real life machines

Model Checking: Motivation

- Traditional methods: Simulation and testing

Model Checking: Motivation

- Traditional methods: Simulation and testing
  - Do not guarantee 100% coverage

Model Checking: Motivation

- Traditional methods: Simulation and testing
  - Do not guarantee 100% coverage
  - Expensive; Labour intensive and time consuming

Model Checking: Motivation

- Traditional methods: Simulation and testing
  - Do not guarantee 100% coverage
  - Expensive; Labour intensive and time consuming
- Can we have methods that guarantee 100% coverage

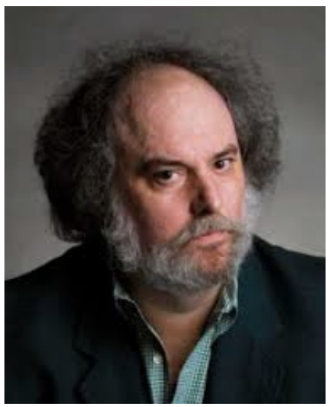
Model Checking: Motivation

- Traditional methods: Simulation and testing
  - Do not guarantee 100% coverage
  - Expensive; Labour intensive and time consuming
- Can we have methods that guarantee 100% coverage
  - Formal Verification: Methods based on Logic and Automata theory

Model Checking: Motivation

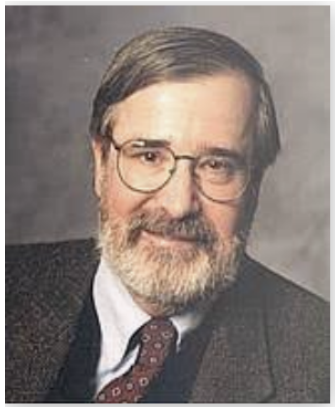
- Traditional methods: Simulation and testing
  - Do not guarantee 100% coverage
  - Expensive; Labour intensive and time consuming
- Can we have methods that guarantee 100% coverage
  - Formal Verification: Methods based on Logic and Automata theory
  - Model Checking is one such popular technique

Model Checking: Cited for Turing Award in 2007 (Invented: 1986)



ACM Turing Award Hon...
cs.utexas.edu

E. Allen Emerson



Introduction to Model C...
moves.rwth-aachen.de

Edmund Clarke

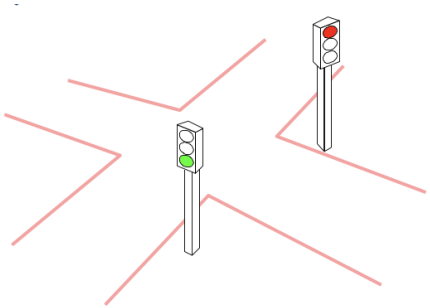
Joseph Sifakis | French com...
britannica.com

Joseph Sifakis

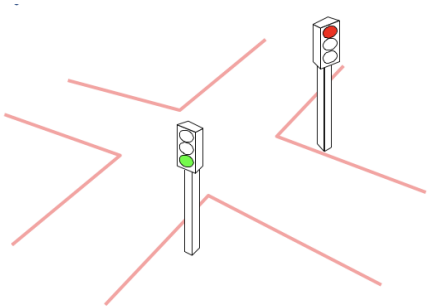
CITATION

Together with E. Allen Emerson and Joseph Sifakis, for their role in developing Model-Checking into a highly effective verification technology that is widely adopted in the hardware and software industries.

What Kind of Systems and Properties?

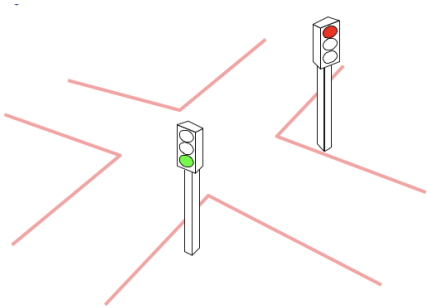


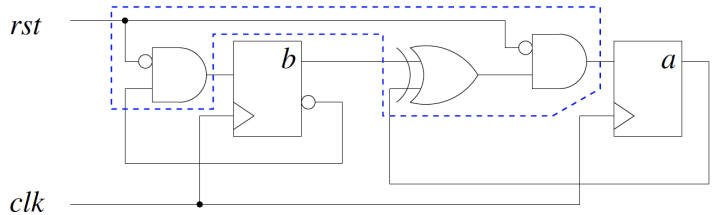
What Kind of Systems and Properties?



"traffic lights should never be green at the same time"

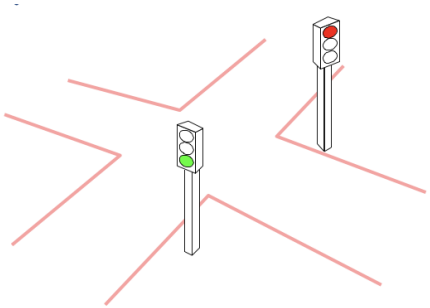
What Kind of Systems and Properties?



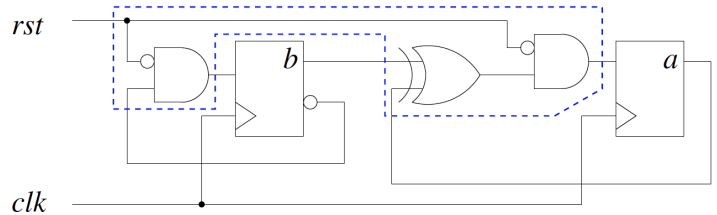


"traffic lights should never be green at the same time"

What Kind of Systems and Properties?

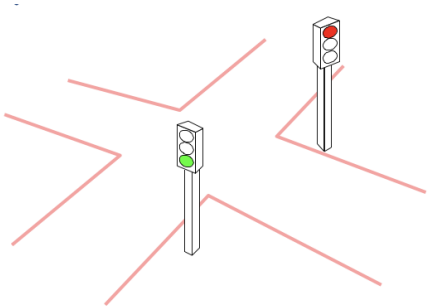


"traffic lights should never be green at the same time"

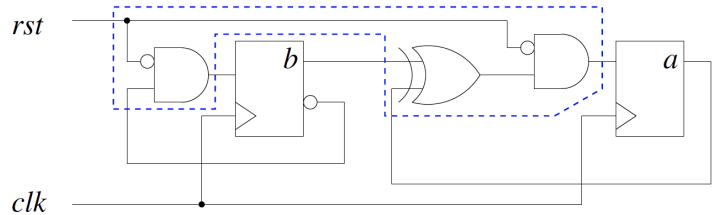


"the counter value ('ab') is always ≤ 3"

What Kind of Systems and Properties?



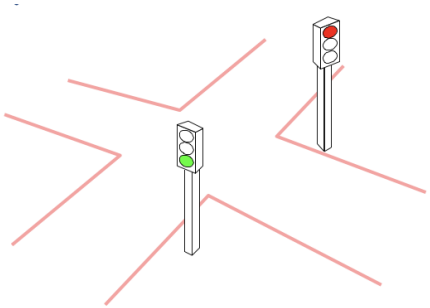
"traffic lights should never be green at the same time"



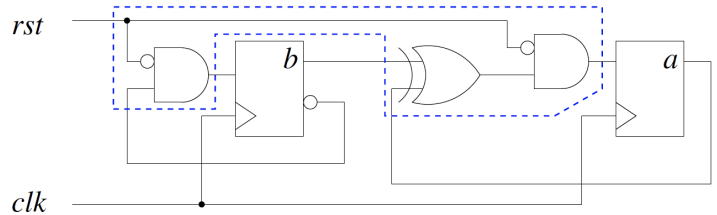
"the counter value ('ab') is always ≤ 3"

```
0: x=0;
1: y=0;
2: while (1) {
3:    x = (x+1) mod 3;
4:    y = x+1; }
```

What Kind of Systems and Properties?



"traffic lights should never be green at the same time"

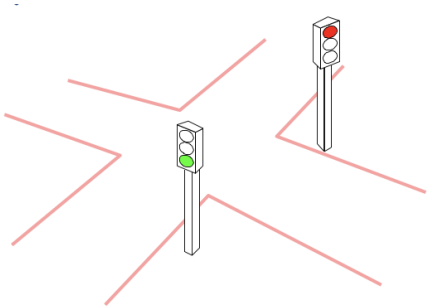


"the counter value ('ab') is always ≤ 3"

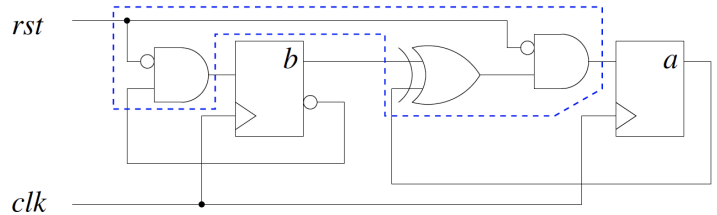
```
0: x=0;
1: y=0;
2: while (1) {
3:    x = (x+1) mod 3;
4:    y = x+1; }
```

"@line 3: $y \geq x$ *always* holds"
"@line 3: x=0 holds infinitely often"

What Kind of Systems and Properties?



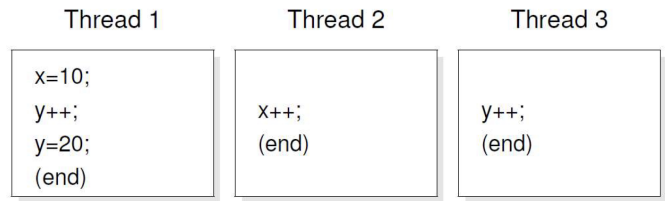
"traffic lights should never be green at the same time"



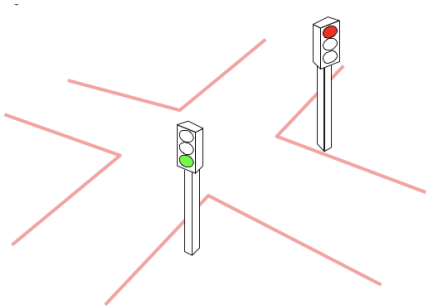
"the counter value ('ab') is always ≤ 3"

```
0: x=0;
1: y=0;
2: while (1) {
3:    x = (x+1) mod 3;
4:    y = x+1; }
```

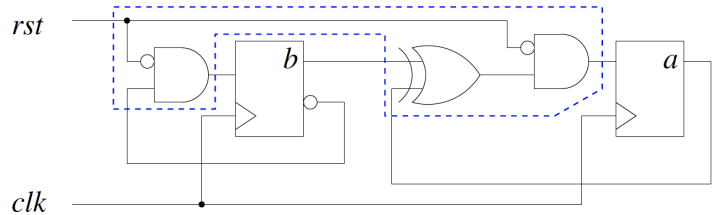
"@line 3: $y \geq x$ *always* holds"
"@line 3: x=0 holds infinitely often"



What Kind of Systems and Properties?



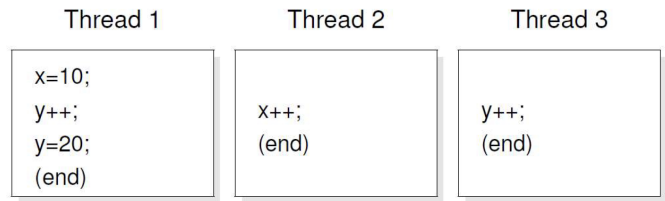
"traffic lights should never be green at the same time"



"the counter value ('ab') is always ≤ 3"

```
0: x=0;
1: y=0;
2: while (1) {
3:    x = (x+1) mod 3;
4:    y = x+1; }
```

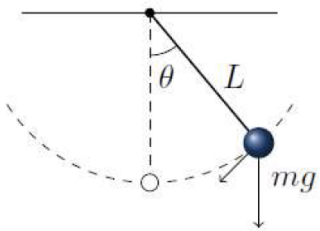
"@line 3: $y \geq x$ *always* holds"
"@line 3: x=0 holds infinitely often"

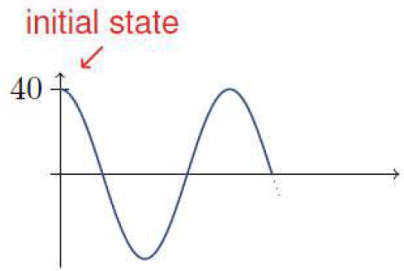


"the program has no data races"

What Kind of Systems Not

An Example





Our model:

$$\frac{\delta^2\theta}{\delta t^2} = -\frac{g}{L}\sin\theta$$

Plot of θ over time

What do we need for formal verification?

- Need a common formalism to model all systems of interest: **Transition Systems (M)**

What do we need for formal verification?

- Need a common formalism to model all systems of interest: **Transition Systems (M)**
- Need an expressive logic/language to specify desired properties **"Special-purpose Temporal Logics (Prop)"**

What do we need for formal verification?

- Need a common formalism to model all systems of interest: **Transition Systems (M)**
- Need an expressive logic/language to specify desired properties **"Special-purpose Temporal Logics (Prop)"**
- Need efficient algorithms to automatically check properties **M** $\models$ **Prop**

Other Powerful Formal Verification Methods

1. Abstract Interpretation (Static Analysis)
   - Automatic, most scalable and widely used
   - Imprecise (False negatives)

Other Powerful Formal Verification Methods

1. Abstract Interpretation (Static Analysis)
   - Automatic, most scalable and widely used
   - Imprecise (False negatives)
2. Theorem Proving
   - Most general: For Models and properties
   - Mostly Interactive

Other Powerful Formal Verification Methods

1. Abstract Interpretation (Static Analysis)
   - Automatic, most scalable and widely used
   - Imprecise (False negatives)
2. Theorem Proving
   - Most general: For Models and properties
   - Mostly Interactive
3. Model Checking
   - Significant industry interest: Intel, ARM, MicrSoft, Google, Facebook
   - A very active research area

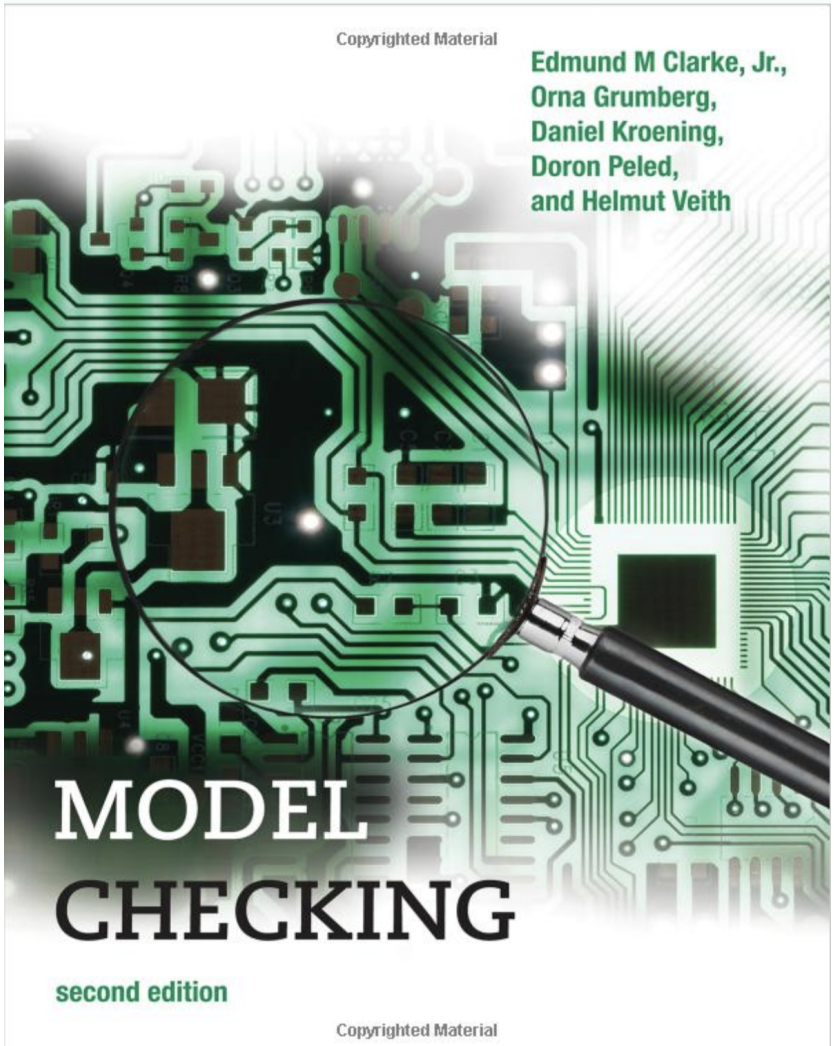
Brief Course Outline

1. Modeling Systems (Kripke Structures)
2. Specifying Properties (Temporal logics: CTL, LTL, CTL*)
3. Verifying Properties (Model Checking for Temporal logics)
4. Tools Exercises: NuSMV, CBMC, SATABS
5. Abstraction and Induction: Scaling Techniques
6. Automata-Based Model Checking

Course Work and Grade Distribution

1. Written Assignments: 30%
2. Tool Exercises: 30%
3. 2 Quizzes: 40%
   - One can be a small programming project

Reference Text Book



eTextBook will be made available for class usage on shared (sign-up sheet) basis

Access details will be shared on Moodle

Industry Recognition Anecdote

What?

> *"Things like even software verification, this has been the Holy Grail of computer science for many decades, but now in some very key areas, for example, driver verification we're building tools that can do* **actual proof** *about the software and how it works in order to guarantee the reliability."*

Bill Gates, April 18, 2002
Keynote address at WinHec 2002