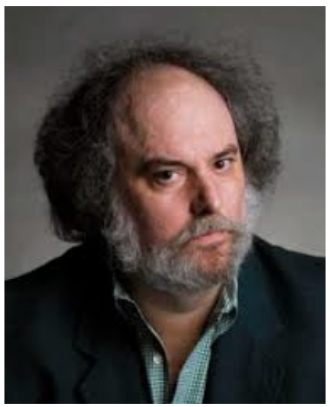
Modeling Module 1

August 13, 2020

Outline

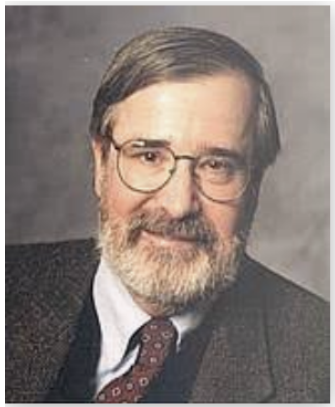
- What's a Model?
- State-Transition Systems (TS)
- Abstraction and Non-determinism in Modeling
- Modeling Examples
- Kripke Structures, Behaviors of TS

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

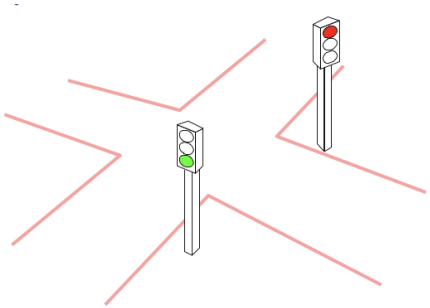
Joseph Sifakis | French com...
britannica.com

Joseph Sifakis

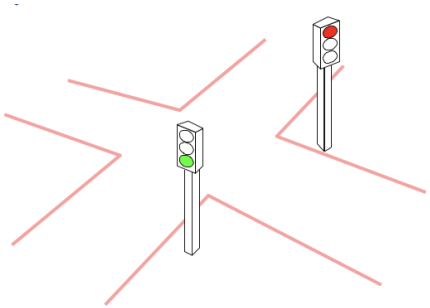
CITATION

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?

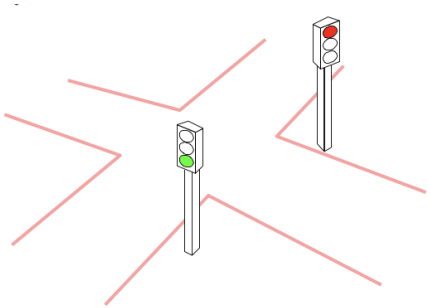


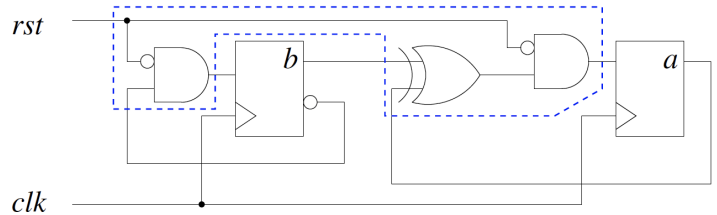
What Kind of Systems and Properties?



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

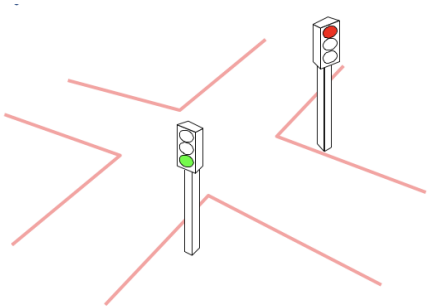
What Kind of Systems and Properties?



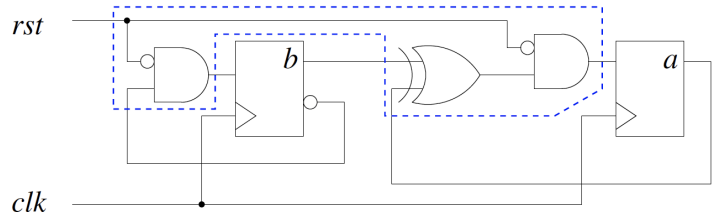


"traffic lights should never be green at the same time"

What Kind of Systems and Properties?

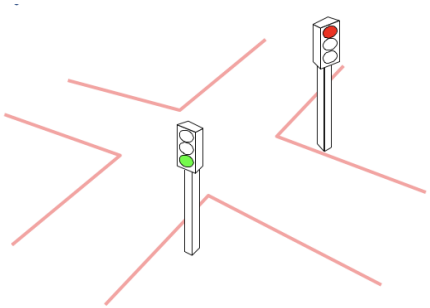


"traffic lights should never be green at the same time"

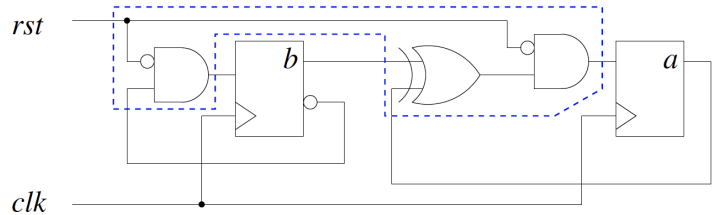


"the counter value ('ab') is always ≤ 2"

What Kind of Systems and Properties?



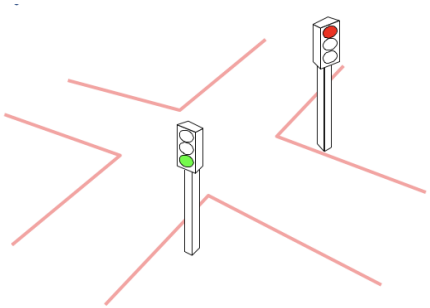
"traffic lights should never be green at the same time"



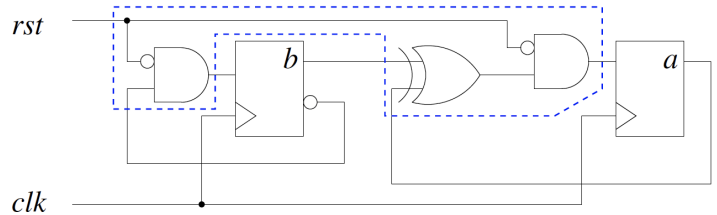
"the counter value ('ab') is always ≤ 2"

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

What Kind of Systems and Properties?



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

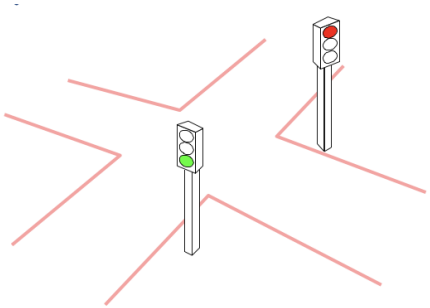


"the counter value ('ab') is always ≤ 2"

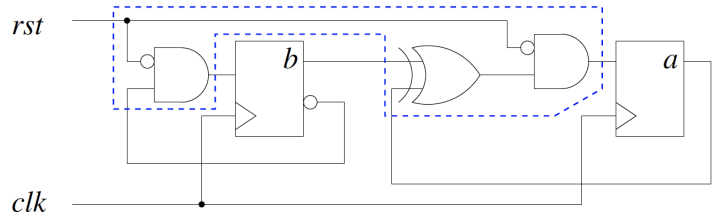
```
0: x=0;
1: y=0;
2: while (1) {
3:    x = (x+1) mod 2;
4:    y = x+1; }
```

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

What Kind of Systems and Properties?



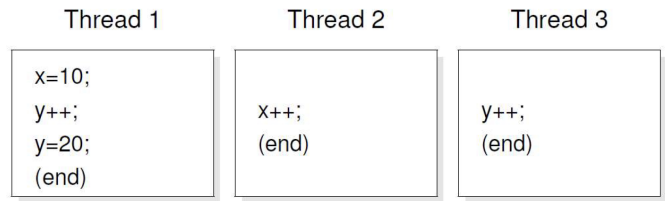
"traffic lights should never be green at the same time"



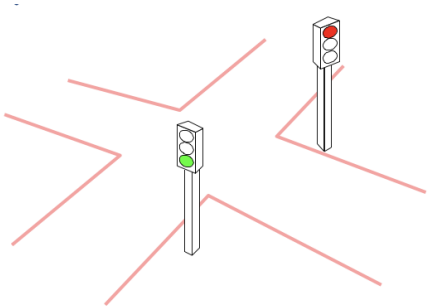
"the counter value ('ab') is always ≤ 2"

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

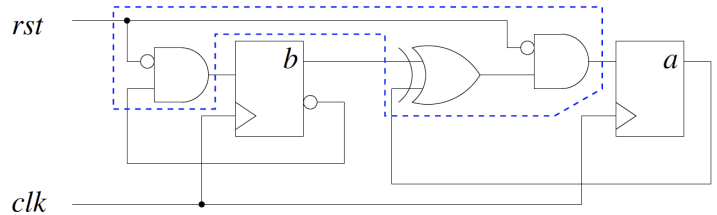
"@line 3: $y \geq x$ *always* holds"
"@line 3: x=0 holds infinitely often"



What Kind of Systems and Properties?



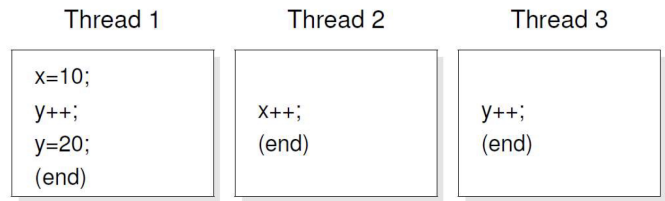
"traffic lights should never be green at the same time"



"the counter value ('ab') is always ≤ 2"

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

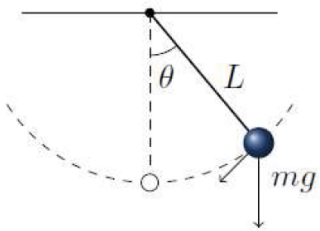
"@line 3: $y \geq x$ *always* holds"
"@line 3: x=0 holds infinitely often"

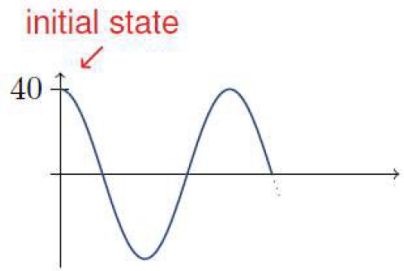


"the program has no data races"

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)"**

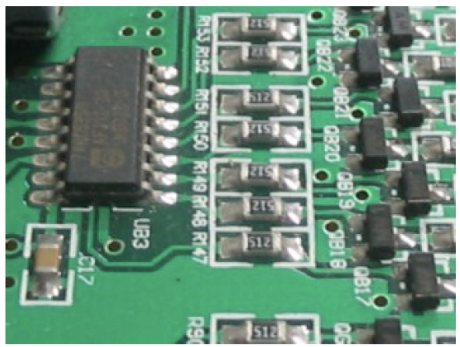
What do we need for formal verification?

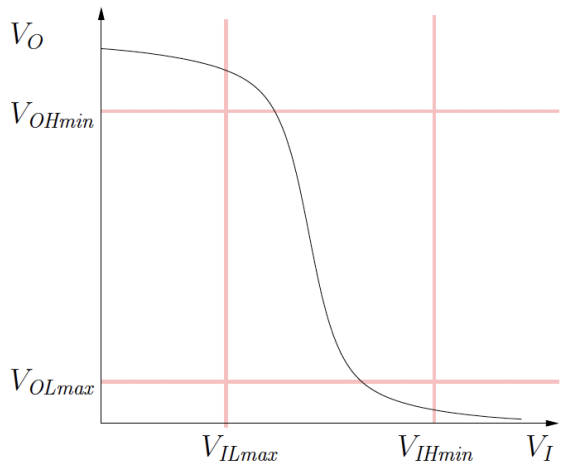
- Need a common formalism to model all systems of interest: **Transition Systems (M)**
- Need an expressive logic/language to specify desired properties **"Special-purpose Temporal Logics (Prop)"**
- Need efficient algorithms to automatically check properties **M** $\models$ **Prop**

A Hardware Abstraction Example

We always omit some detail.

Example:





x	$\neg x$
0	1
1	0

Abstraction

Modeling: States

- Our systems have a state,
  the model has a set of states

Modeling: States

- Our systems have a state,
  the model has a set of states
- This is typically given as a valuation of state variables
  - θ
  - Velocity
  - Data stored in program variables
  - Data stored in hardware registers/flip-flops
  - ....

Modeling: States

- Our systems have a state,
  the model has a set of states
- This is typically given as a valuation of state variables
  - θ
  - Velocity
  - Data stored in program variables
  - Data stored in hardware registers/flip-flops
  - ....
- We focus on discrete state variables

Modeling State Transitions

- When the state changes with time, we have a state transition

Modeling State Transitions

- When the state changes with time, we have a state transition
- Time can be modeled as
  - continuous
  - discrete

Modeling State Transitions

- When the state changes with time, we have a state transition
- Time can be modeled as
  - continuous
  - discrete
- We will focus on discrete-time transitions

Definition Transition System

Definition (Transition System)
A *Transition System* is a 3-tuple $M = (S, S_0, T)$ consisting of

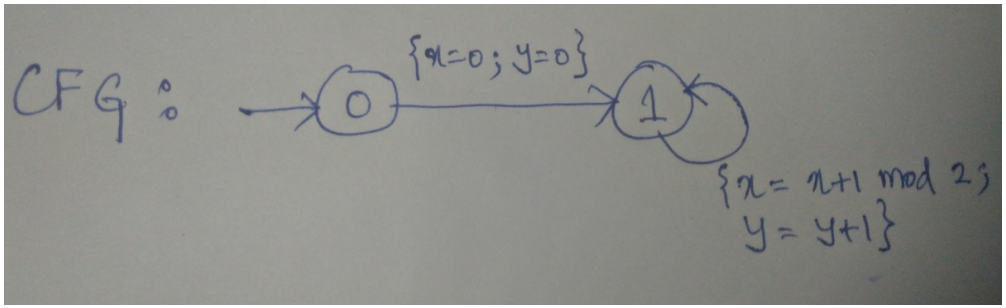
- a set of states S,
- a set of initial states $S_0 \subseteq S$
- a transition relation $T \subseteq (SXS)$

Modeling Sequential Code: An Example

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

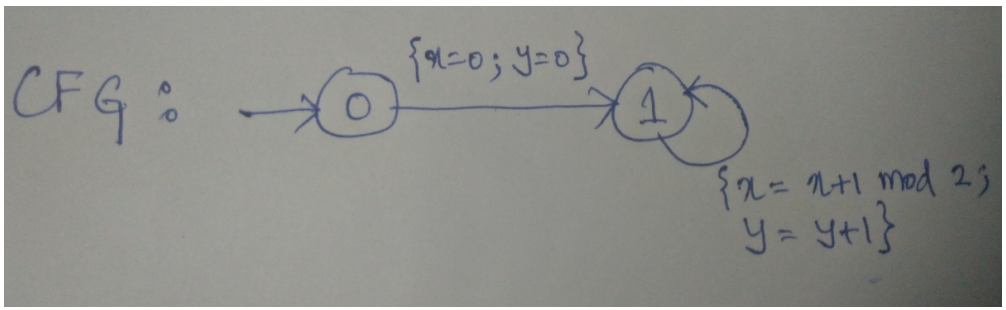
Modeling Sequential Code: An Example

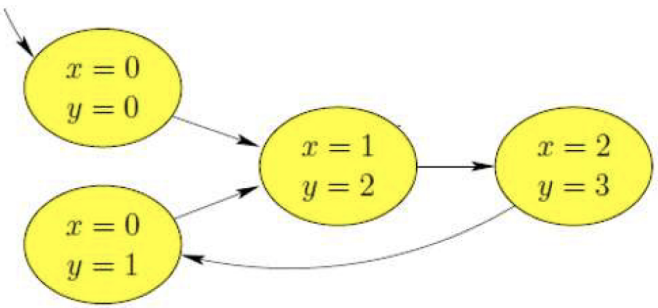
```
0: x=0;
1: y=0;
2: while (1) {
3:   x = (x+1) mod 2;
4:   y = x+1; }
```



Modeling Sequential Code: An Example

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





TS : $M = (S, S_0, R)$

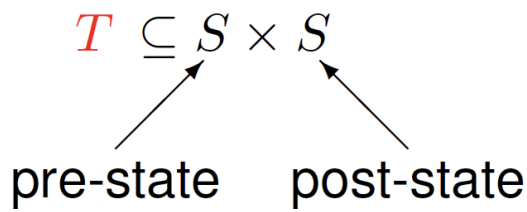
- $S = (D_1 X D_2)$, where $D_1 = \{0, 1, 2\}$, $D_2 = \{0, 1, 2, 3\}$
- $S_0 = \{(0, 0)\}$
- $R = \{((0, 0), (1, 2)), ((0, 1), (1, 2)), ((1, 2), (2, 3)), ((2, 3), (0, 1))\}$

Towards Symbolic Representation for TS: Notation for States

- S - set of states
- $s \in S$ - one particular state
- s, x - the value of some variable x in state s.
  These are called state variables

Towards Symbolic Representation for TS: The Transition Relation

The transition relation T relates a pre-state to a post-state:



The transition relation captures how the state can be transformed by a single transition.

Towards Symbolic Representation for TS: Characteristic Functions

Notation: We typically write the transition relation using a characteristic function.

This is useful for any kind of set:

$$S = \{0, 2, 4, 6, 8, \ldots\} \qquad x \in S \iff x \bmod 2 = 0$$

$$S = \{1, 2, \ldots, 9, 10\} \qquad x \in S \iff 1 \leq x \wedge x \leq 10$$

$$S = \{1, 2, 4, 8, 16, 32, \ldots\} \qquad x \in S \iff \exists y.\, x = 2^y$$

Towards Symbolic Representation for TS: The Transition Relation Again

Example:

$$T \subseteq \mathbb{N}_0 \times \mathbb{N}_0$$

$$T(x, x') \iff x' = x + 1$$

This corresponds to the set

$$\{(0, 1), (1, 2), (2, 3), \ldots\}$$

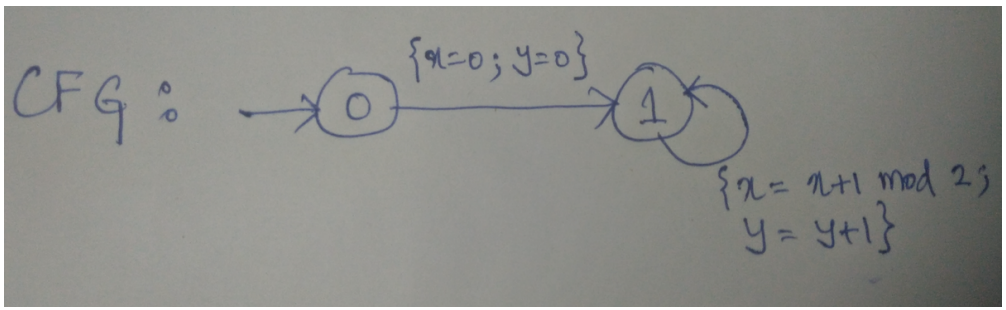
By convention, primed variables (x') denote the value of a state variable in the next state.

Symbolic TS for the Code Example

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

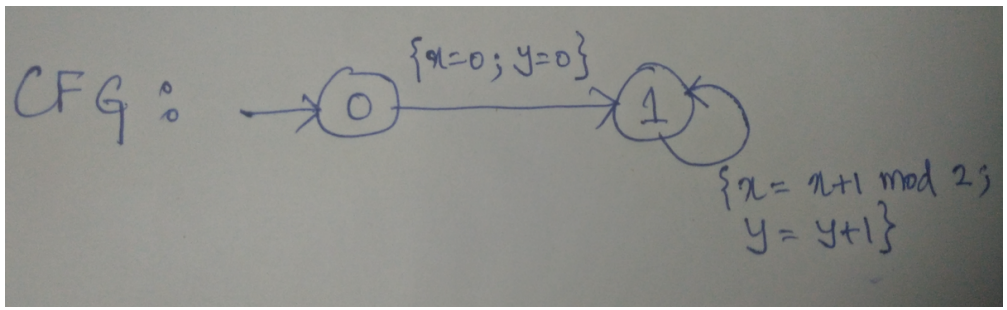
Symbolic TS for the Code Example

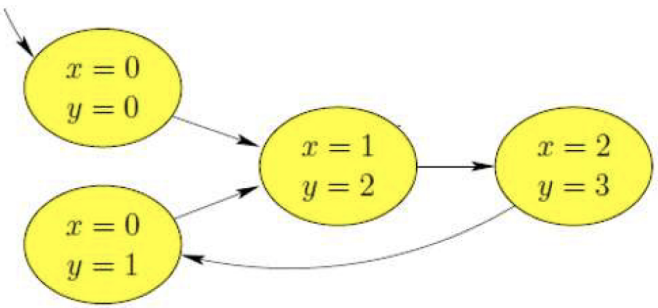
```
0: x=0;
1: y=0;
2: while (1) {
3:    x = (x+1) mod 2;
4:    y = x+1; }
```



Symbolic TS for the Code Example

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





TS : $M = (S, S_0, R)$

- $S = (D_1 X D_2)$, where $D_1 = \{0, 1, 2\}$, $D_2 = \{0, 1, 2, 3\}$
- $S_0(s) : (s.x = 0 \wedge s.y = 0)$
- $R(s, s') : (s'.x = s.x + 1 \ mod \ 3) \wedge (s'.y = s'.x + 1)$

Role of Nondeterminism in Modeling

- *Nondeterminism* is used to abstract, i.e., not deliberately committing, to certain details in modeling systems

Role of Nondeterminism in Modeling

- *Nondeterminism* is used to abstract, i.e., not deliberately committing, to certain details in modeling systems
  - Initial nondeterminism: Initial state need not be unique

Role of Nondeterminism in Modeling

- *Nondeterminism* is used to abstract, i.e., not deliberately committing, to certain details in modeling systems
  - Initial nondeterminism: Initial state need not be unique
  - Inputs (environment) nondeterminism: Inputs can change arbitrarily

Role of Nondeterminism in Modeling

- *Nondeterminism* is used to abstract, i.e., not deliberately committing, to certain details in modeling systems
  - Initial nondeterminism: Initial state need not be unique
  - Inputs (environment) nondeterminism: Inputs can change arbitrarily
  - Scheduler nondeterminism (concurrent systems): Multiple successor states possible for a transition

Initial State Nondeterminism

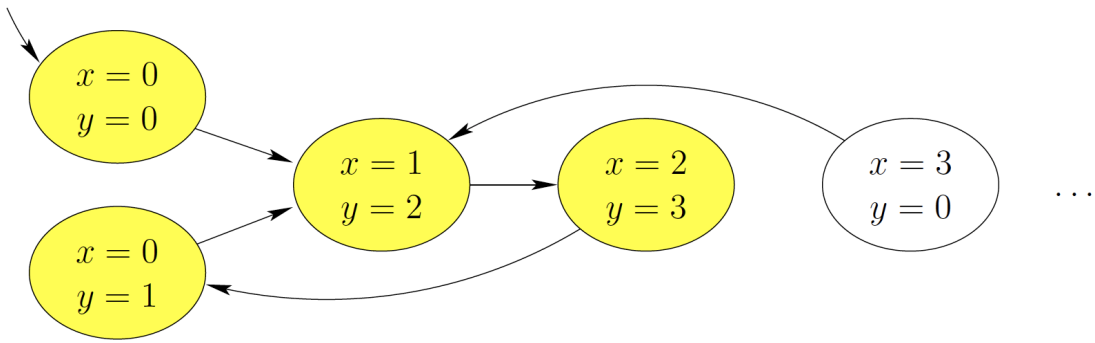
```
0: while (1) {
1:   x = (x+1) mod 2;
2:   y = x+1; }
```

Initial State Nondeterminism

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

Initial State Nondeterminism

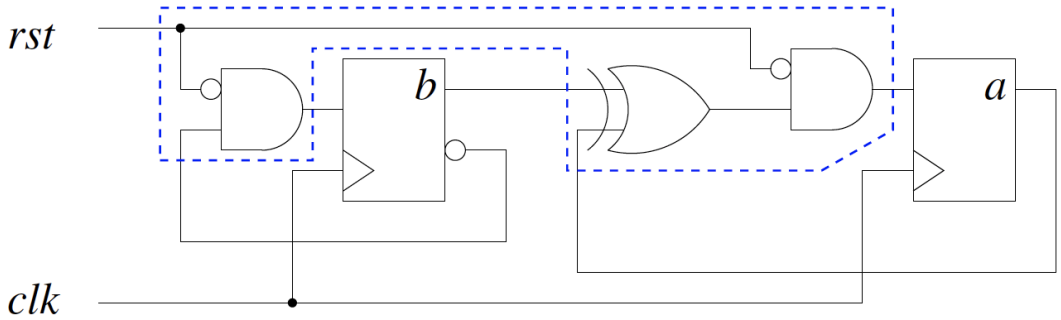
```
0: while (1) {
1:    x = (x+1) mod 2;
2:    y = x+1; }
```



$TS: \ M = (S, S_0, R)$

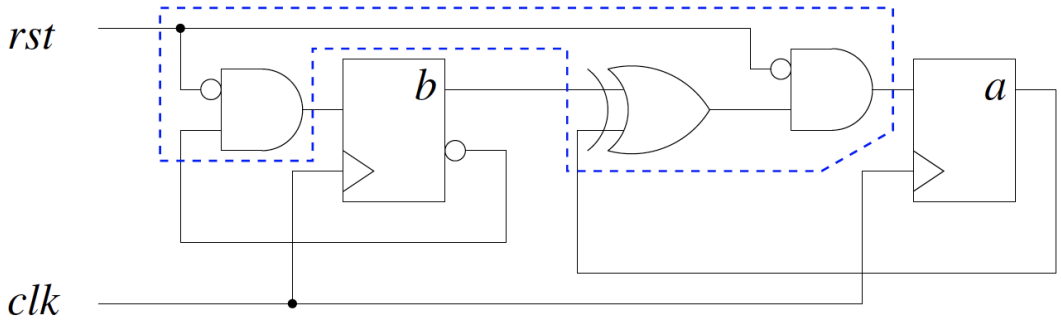
- $S = (Nat \ X \ Nat)$
- $S_0(s) : (s.y = 0)$
- $R(s, s') : (s'.x = s.x + 1 \ mod \ 3) \wedge (s'.y = s'.x + 1)$

Modeling Clocked Hardware: Modulo 4 Counter



- One global clock: *clk*
- One input: *rst*
- Next-state logic (dashed box):
  a, b (flip-flops) change *synchronously* with every tick of *clk*
  - $\delta(a, b, rst).a \equiv \overline{rst} \ . \ (a \oplus b)$
  - $\delta(a, b, rst).b \equiv \overline{rst} \ . \ \overline{b}$

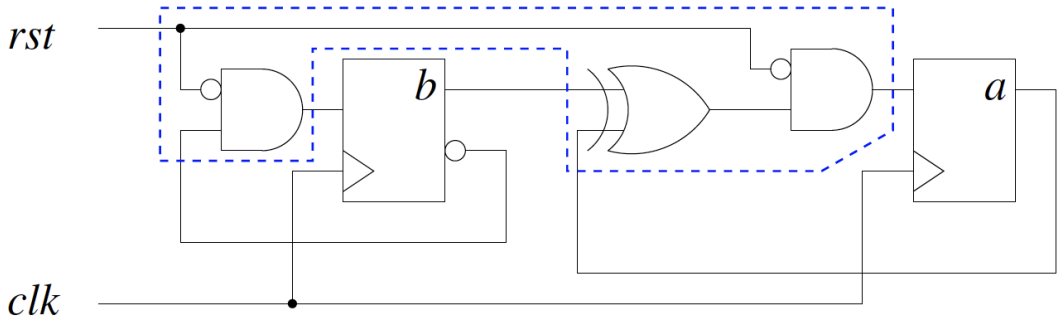
Modeling Inputs in TS

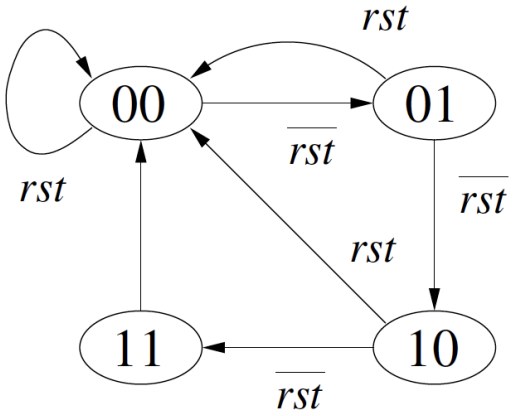


$$T(s, s') \equiv$$

$$(\delta(s.a, s.b, s.rst) = s'.ab \vee$$

Modeling Inputs in TS

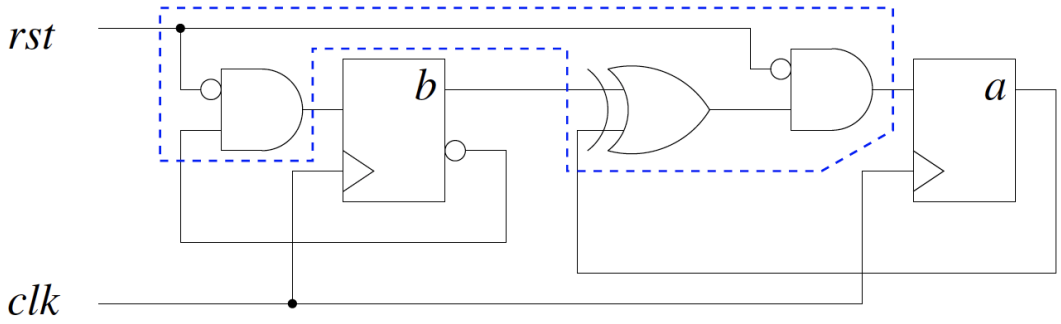


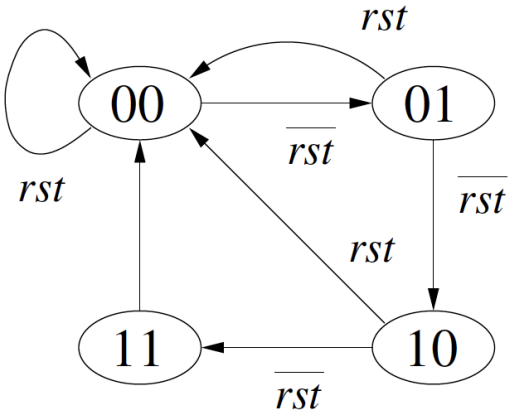


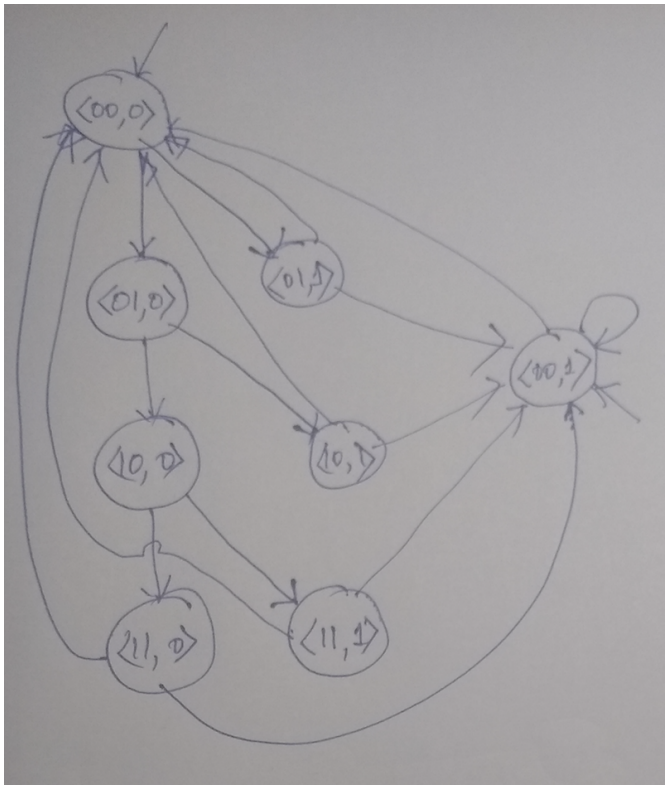
$$T(s, s') \equiv$$

$$(\delta(s.a, s.b, s.rst) = s'.ab \vee$$

Modeling Inputs in TS







$$T(s,s') \equiv$$

$$(\delta(s.a, s.b, s.rst) = s'.ab \vee$$

Assumptions on Transition Systems

Definition (Transition System)
A *Transition System* is a 3-tuple $M = (S, S_0, T)$ consisting of

- a set of states S,
- a set of initial states $S_0 \subseteq S$
- a transition relation $T \subseteq (SXS)$

1. Domains of state variables are finite and discrete

Assumptions on Transition Systems

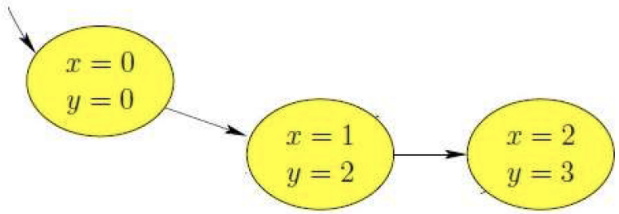
Definition (Transition System)
A *Transition System* is a 3-tuple $M = (S, S_0, T)$ consisting of

- a set of states S,
- a set of initial states $S_0 \subseteq S$
- a transition relation $T \subseteq (SXS)$

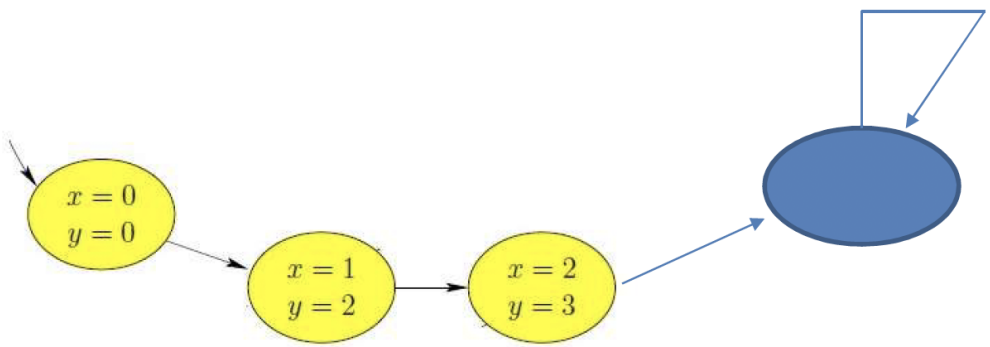
1. Domains of state variables are finite and discrete
2. Transition relation is total (No loss of generality)
   - How about terminating systems, e.g., code?
   - If $T(s, s') \equiv \mathit{false}$, then redfeine it as $T'(s, s') \equiv (s = s')$

Terminating TS

```
Void main () {

int x = 0, y = 0;

while (y<=2) {

x = x+1 mod 3;
y = x+1; }

}
```



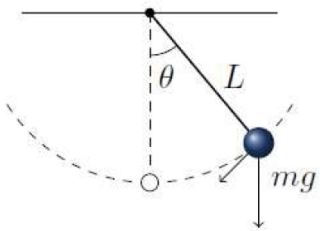
Without loss of generality we make
all our TS non-terminating

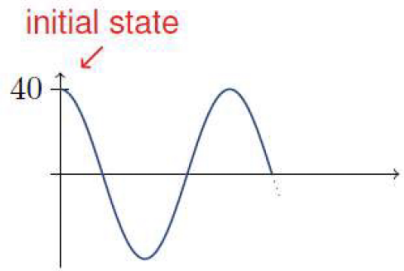


This means our TS relation is assumed to be TOTAL

An Continuous Domain Example

An Example





Our model:

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

Plot of θ over time