Modeling Module 1

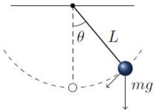
August 12, 2020

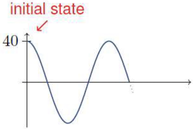
Outline

- What's a Model?
- State-Transition Systems (Kripke Strctures)
- Behaviors, Properties, and the Model Checking Problem
- Modeling and Abstraction
- Modeling Examples

An Continuous Domain Example

An Example





Our model:

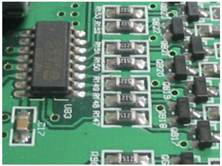
$$\frac{\delta^2\theta}{\delta t^2} = -\frac{g}{L}\sin\theta$$

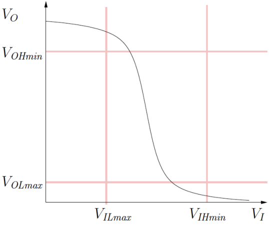
Plot of θ over time

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 <u>discrete state variables</u>

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

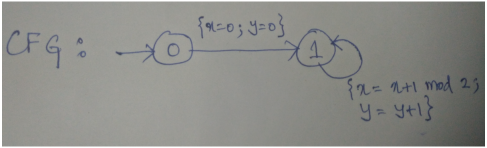
- a set of states S,
- a set of initial states $S_0 \subseteq S$
- a transition relation $T \subseteq (SXS)$

Modeling Sequential Code: An Example

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

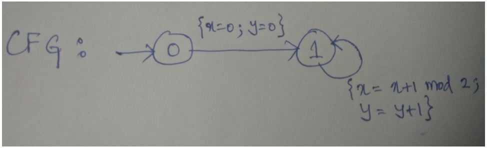
Modeling Sequential Code: An Example

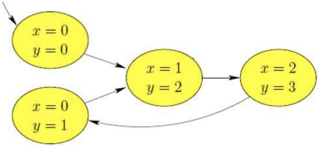
```
0: x=0;
1: y=0;
2: while (1) {
3:   x = (x+1) mod 2;
4:   y = x+1; }
```



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 <u>state variables</u>

Towards Symbolic Representation for TS: The Transition Relation

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

$$T \subseteq \underset{\text{pre-state}}{S} \times \underset{\text{post-state}}{S}$$

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:

$$
\begin{array}{lll}
S = \{0, 2, 4, 6, 8, \ldots\} & & x \in S \iff x \bmod 2 = 0 \\
S = \{1, 2, \ldots, 9, 10\} & & x \in S \iff 1 \leq x \land x \leq 10 \\
S = \{1, 2, 4, 8, 16, 32, \ldots\} & & x \in S \iff \exists y.\, x = 2^y
\end{array}
$$

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

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

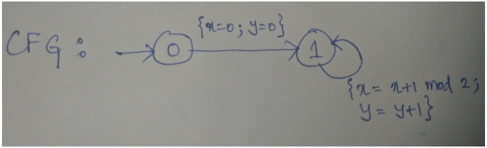
By convention, primed variables (x') denote the value of a state variable in the next state.

Symbolic TS for the Code Example

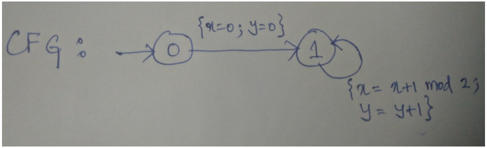
```
0: x=0;
1: y=0;
2: while (1) {
3:   x = (x+1) mod 2;
4:   y = x+1; }
```

Symbolic TS for the Code Example

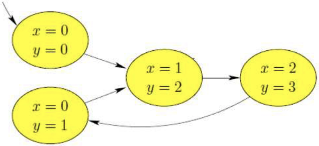
```
0: x=0;
1: y=0;
2: while (1) {
3:   x = (x+1) mod 2;
4:   y = x+1; }
```



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

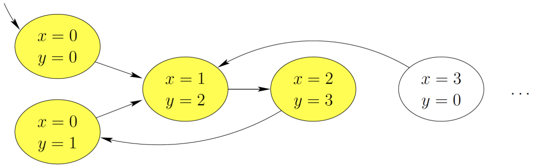
```
0: while (1) {
1:   x = (x+1) mod 2;
2:   y = x+1; }
```

Initial State Nondeterminism

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

Initial State Nondeterminism

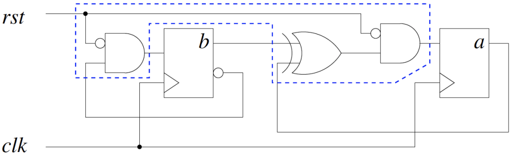
```
0: while (1) {
1:   x = (x+1) mod 2;
2:   y = x+1; }
```



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

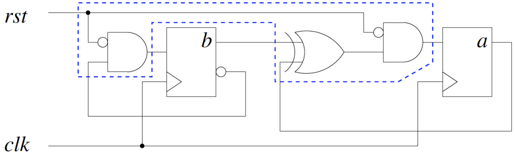
- $S = (Nat \; X \; Nat)$
- $S_0(s) : (s.y = 0)$
- $R(s, s') : (s'.x = s.x + 1 \; mod \; 3) \wedge (s'.y = s'.x + 1)$

Modeling Clocked Hardware

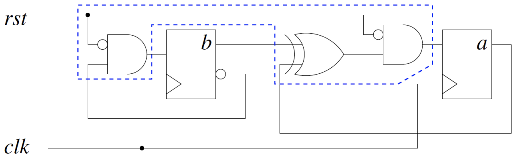


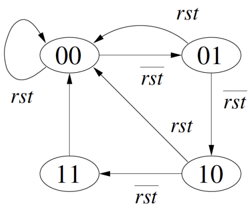
- One global clock: *clk*
- One input: *rst*
- Next-state logic: dashed box

Modeling Inputs in TS



Modeling Inputs in TS





Modeling Inputs in TS