

Modeling Systems

Chapter 3

- Exercise handout today

Modeling Systems

- 3.1 Transition Systems and Kripke Structures
- 3.2 Nondeterminism and Inputs
- 3.3 First-Order Logic and Symbolic Representations
- 3.4 Boolean Encoding
- 3.5 Modeling Digital Circuits
- 3.6 Modeling Programs
- 3.7 Fairness

Systems and Correctness

- We consider a broad range of systems
 - Hardware (digital circuitry)
 - Software
- We want to check that the system is **correct**
 - Meets high-level requirements
 - Captured in the form of **system properties**

Why Model?

Specification

States what you want to prove

System

Abstract away unnecessary details

- How does the OS scheduler work?
- How is the CPU pipeline implemented?
- What are the voltage levels in the CPU?

But careful!

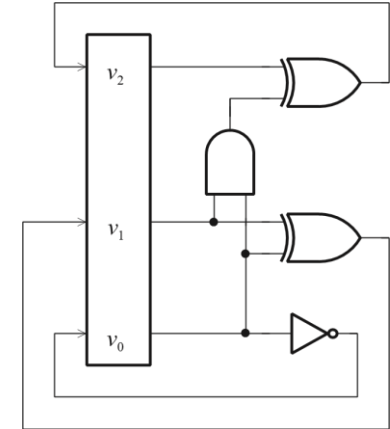
- Carelessly implemented CPUs introduce side channels
- Alpha particles may cause bits to flip
- Your formally verified system will fail when hit with a hammer
- ...

What is a Model?

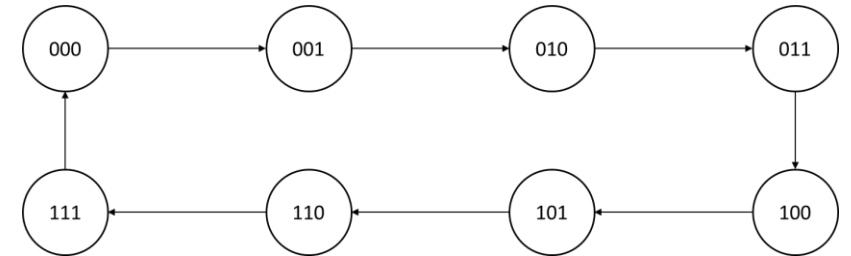
- A **model** is a description of the **behavior** of the system
- Behavior is
 - a set of observations
 - as the system evolves its state over time
- We check **algorithmically** that the model satisfies the properties
- To this end the model...
 - must have **sufficient detail** to prove the property
 - but should not be **too complex**

Three Models

- Circuits



- Kripke Structures



- Characteristic Functions

$$\begin{aligned}\mathcal{R}_0(V, V') &= (v'_0 \leftrightarrow \neg v_0) \\ \mathcal{R}_1(V, V') &= (v'_1 \leftrightarrow v_0 \oplus v_1) \\ \mathcal{R}_2(V, V') &= (v'_2 \leftrightarrow v_2 \oplus (v_0 \wedge v_1)) \\ \mathcal{R}(V, V') &= \mathcal{R}_0 \wedge \mathcal{R}_1 \wedge \mathcal{R}_2\end{aligned}$$

Kripke Structures

Inputs: Light Switch Example

- Input: “button pressed” or “button released”, controlled by a hand, which is part of the environment
- Output: “light on” or “light off”



- Button is “retractive”, it bounces back
- When the light is off, pushing the button turns the light on
- When the light is on, pushing the button turns the light off

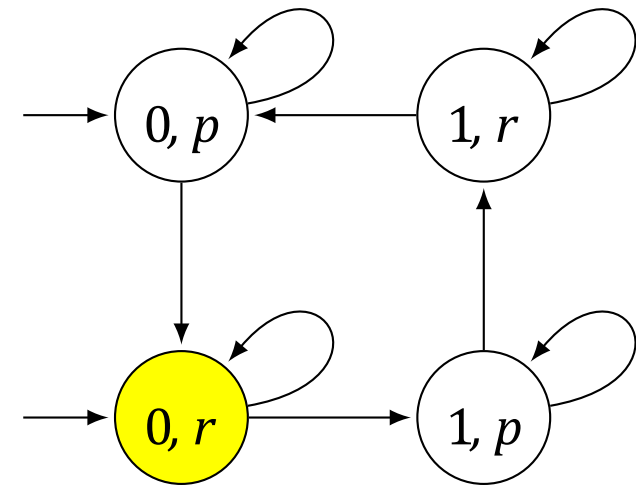
Inputs : Light Switch Example



light switch
"released" = r



light bulb
"off" = 0



model of the controller

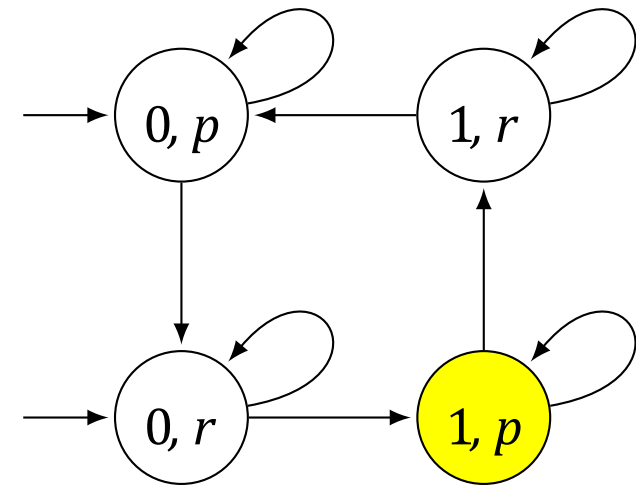
Inputs : Light Switch Example



light switch
"pressed" = p



light bulb
"on" = 1



model of the controller

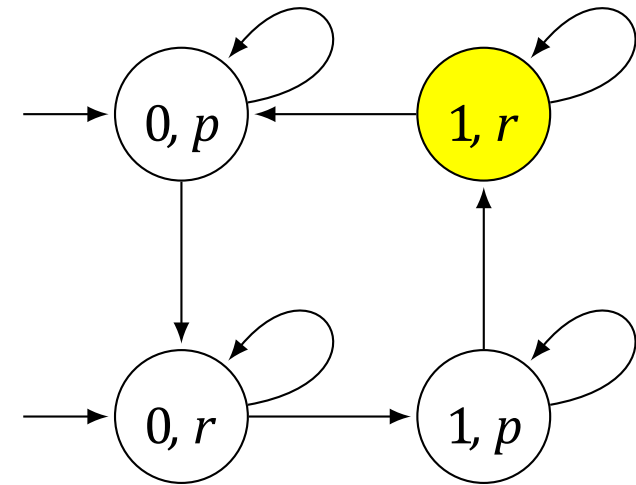
Inputs : Light Switch Example



light switch
"released" = r



light bulb
"on" = 1



model of the controller

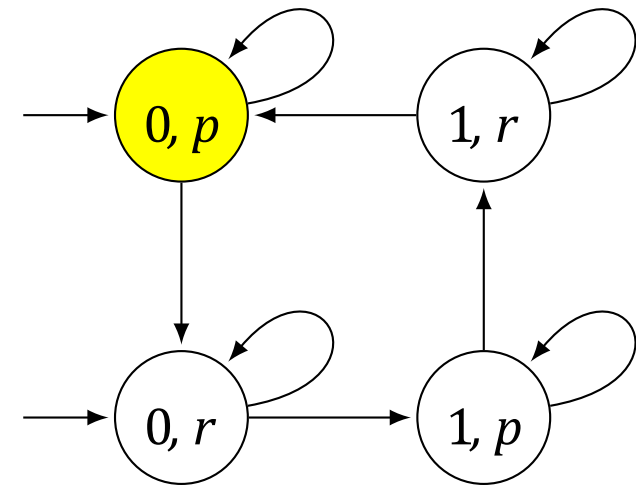
Inputs : Light Switch Example



light switch
"pressed" = p



light bulb
"off" = 0



model of the controller

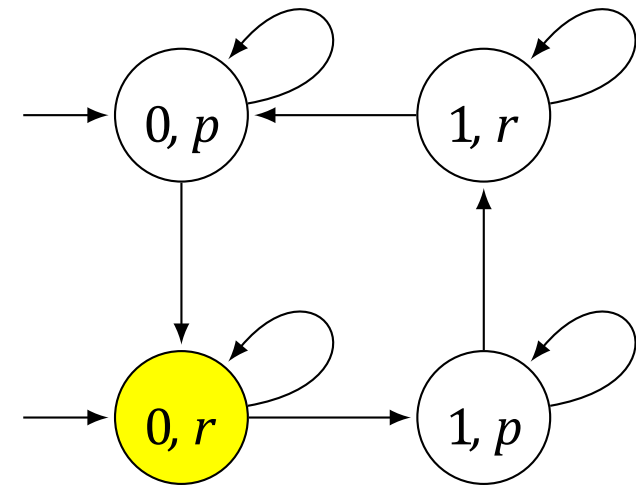
Inputs : Light Switch Example



light switch
"released" = r



light bulb
"off" = 0



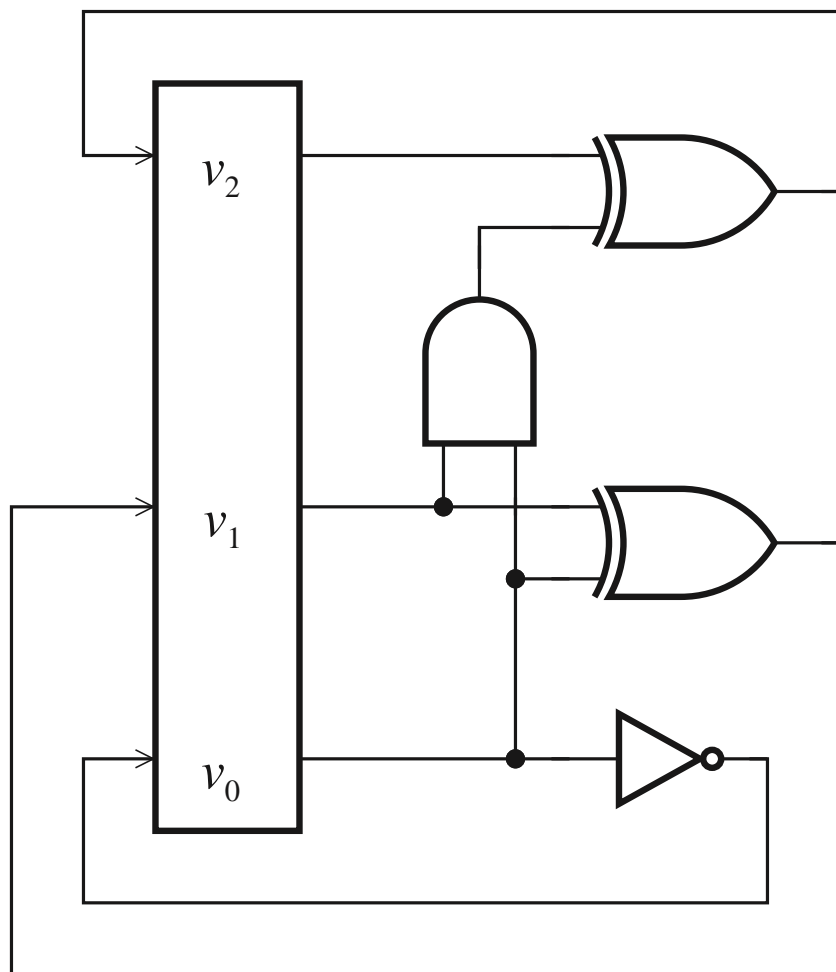
model of the controller

Kripke Structure $M = (S, S_0, R, AP, L)$

- S – (finite) set of **states**
- $S_0 \subseteq S$ – set of **initial states**
- $R \subseteq S \times S$ – left-total **transition relation**
 - For every $s \in S$ there exists $s' \in S$ such that $(s, s') \in R$
 - Left-total implies that every path is infinite
- AP – finite set of **atomic propositions**
- $L : S \rightarrow 2^{AP}$ – **labeling function** that associates every state with the atomic propositions true in that state. *We include inputs (if we are interested in them)*

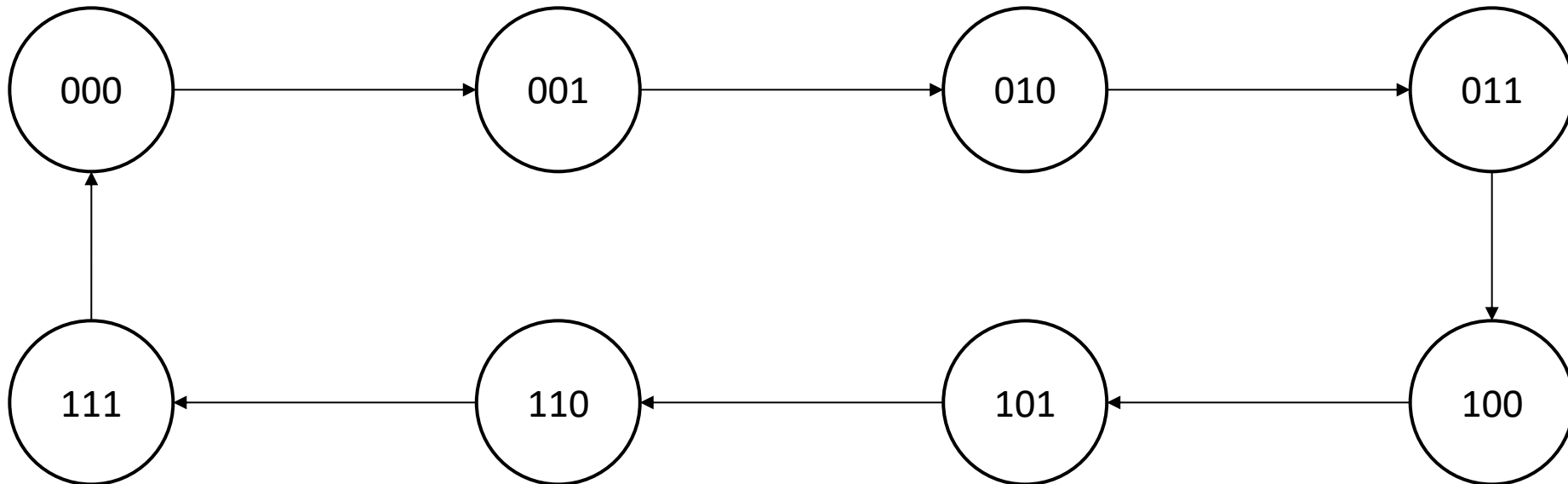
First-Order Logic and Symbolic Representations

3-bit Counter



$$\begin{aligned}\mathcal{R}_0(V, V') &= (v'_0 \leftrightarrow \neg v_0) \\ \mathcal{R}_1(V, V') &= (v'_1 \leftrightarrow v_0 \oplus v_1) \\ \mathcal{R}_2(V, V') &= (v'_2 \leftrightarrow v_2 \oplus (v_0 \wedge v_1)) \\ \mathcal{R}(V, V') &= \mathcal{R}_0 \wedge \mathcal{R}_1 \wedge \mathcal{R}_2\end{aligned}$$

Kripke Structure



Symbolic Representation

$$V = \{v_1, \dots, v_n\}$$

system variables

$$D_v$$

domain of v

$$s: V \rightarrow \prod_{v \in V} D_v$$

valuation, state

Example

Symbolic Representation

$V = \{v_1, \dots, v_n\}$ system variables

D_v domain of v

$s: V \rightarrow \prod_{v \in V} D_v$ valuation, state

Example

$V = \{v_1, v_2, v_3\}, D_{v_i} = \mathbf{N}$

State space: \mathbf{N}^V (or simply \mathbf{N}^3)

examples of state: $\{(v_1, 2), (v_2, 3), (v_3, 8)\}$ (short: $(2,3,8)$)

Characteristic Functions

In general, a formula is a set of states.

Characteristic Functions

In general, a formula is a set of states.

$$v_1 = 2 \wedge v_2 = 3 \wedge v_3 = 8$$

$$(2,3,8)$$

$$v_1 = 2 \wedge v_2 = 3$$

$$\{(2,3,n_3) \mid n_3 \in \mathbf{N}\}$$

$$v_2 = 3 \wedge v_3 = v_1 + v_2$$

$$\{(n_1, 3, n_1 + 3) \mid n_1 \in \mathbf{N}\}$$

true

$$\mathbf{N}^3$$

Sets and Formulas

Formula

\mathcal{A}, \mathcal{B}

Set

A, B

$A \cup B$

$A \cap B$

$S = D_{v_1} \times \dots \times D_{v_n}$

$S \setminus A$

Example

$$v_1 = 2 \wedge v_2 = 3$$

$$v_2 = 3 \wedge v_3 = v_1 + v_2$$

$$\{(2, 3, n_3) \mid n_3 \in \mathbf{N}\}$$

$$\{(n_1, 3, n_1 + 3) \mid n_1 \in \mathbf{N}\}$$

•

Sets and Formulas

Formula	Set
\mathcal{A}, \mathcal{B}	A, B
$\mathcal{A} \vee \mathcal{B}$	$A \cup B$
$\mathcal{A} \wedge \mathcal{B}$	$A \cap B$
$true$	$S = D_{v_1} \times \dots \times D_{v_n}$
$\neg \mathcal{A}$	$S \setminus A$

Example

$$v_1 = 2 \wedge v_2 = 3$$

$$\{(2, 3, n_3) \mid n_3 \in \mathbf{N}\}$$

$$v_2 = 3 \wedge v_3 = v_1 + v_2$$

$$\{(n_1, 3, n_1 + 3) \mid n_1 \in \mathbf{N}\}$$

$$v_1 = 2 \wedge v_2 = 3 \wedge v_2 = 3 \wedge v_3 = v_1 + v_2$$

$$(2, 3, 5)$$

$$v_1 = 2 \wedge v_2 = 3 \vee v_2 = 3 \wedge v_3 = v_1 + v_2$$

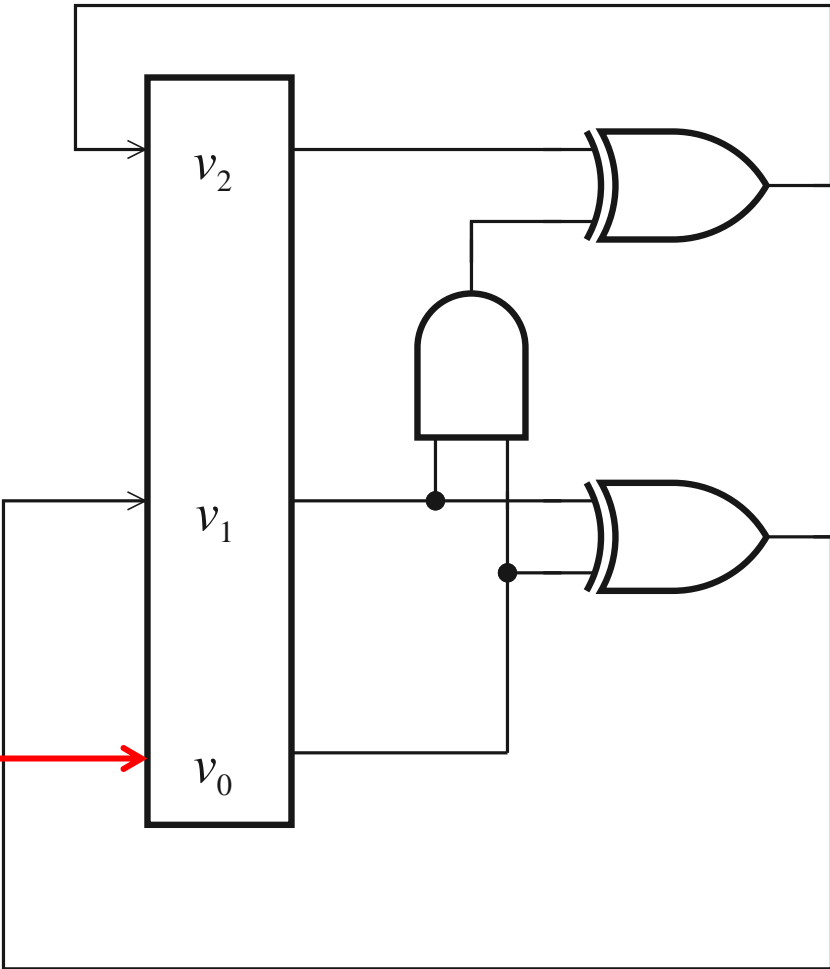
$$\{(2, 3, n_3) \mid n_3 \in \mathbf{N}\} \cup \{(n_1, 3, n_1 + 3) \mid n_1 \in \mathbf{N}\}$$

Inputs

Inputs can be anything - model as nondeterministic

$$\mathcal{R}_0(V, V') =$$

$$\mathcal{R}_1(V, V') = (v'_1 \leftrightarrow v_0 \oplus v_1) \quad \mathcal{R}_2(V, V') = (v'_2 \leftrightarrow v_2 \oplus (v_0 \wedge v_1))$$



Inputs

Inputs can be anything - model as nondeterministic

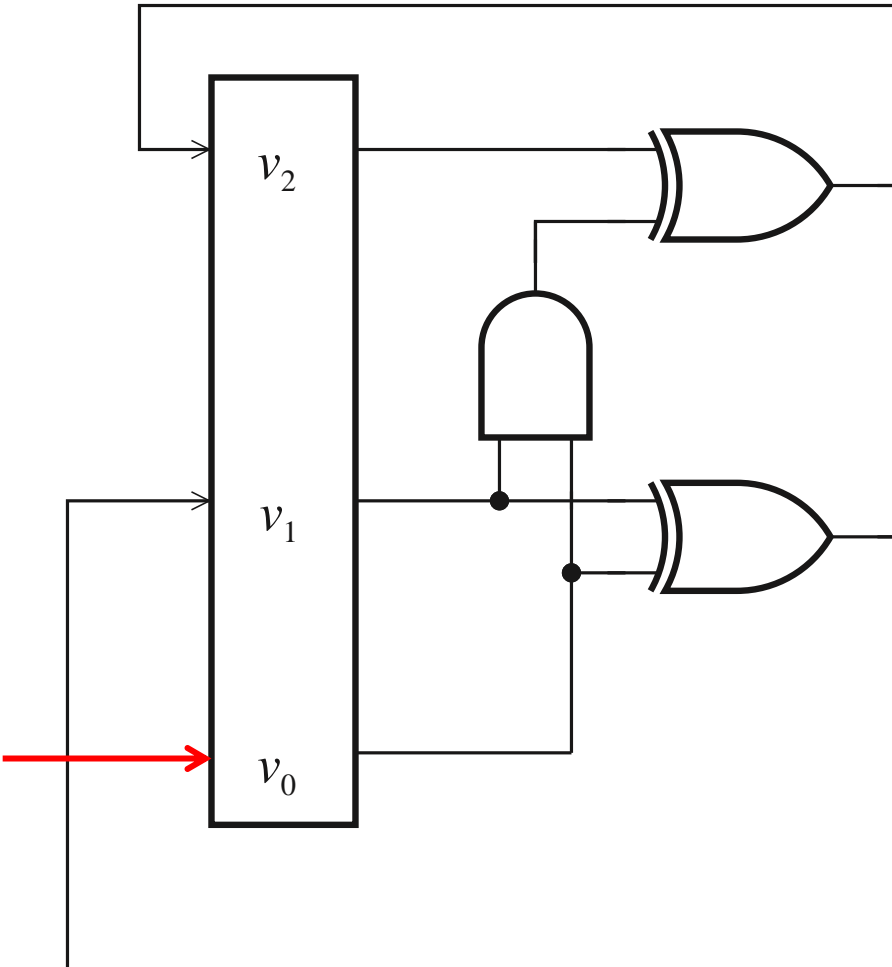
$$\mathcal{R}_0(V, V') = \text{true} \text{ no constraints on } v_1$$

$$\mathcal{R}_1(V, V') = (v'_1 \leftrightarrow v_0 \oplus v_1)$$

$$\mathcal{R}_2(V, V') = (v'_2 \leftrightarrow v_2 \oplus (v_0 \wedge v_1))$$

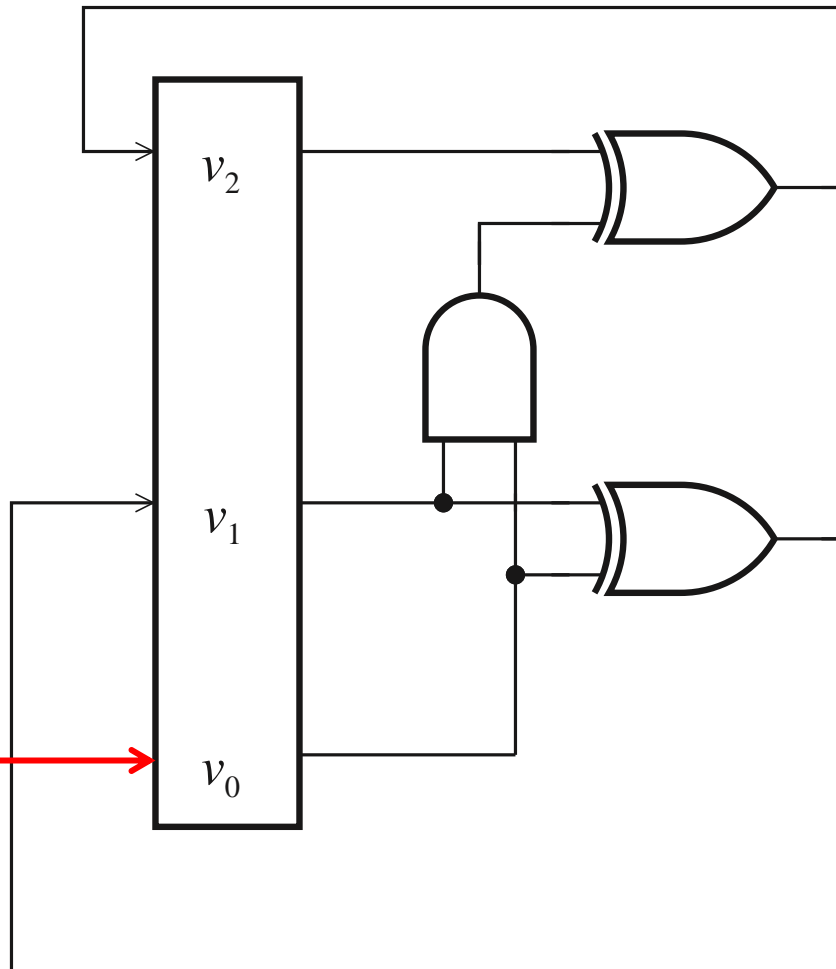
$$\mathcal{R}(V, V') = \mathcal{R}_1(V, V') \vee \mathcal{R}_2(V, V')$$

What is the Kripke structure?



Inputs

What is the Kripke structure?



Properties

Can I say “if the state is 000, the next state is 001”?

Properties

Can I say “if the state is 000, the next state is 001”?

$$\neg v_2 \wedge \neg v_1 \wedge \neg v_0 \wedge \mathcal{R}(V, V') \rightarrow \neg v'_2 \wedge \neg v'_1 \wedge v'_0$$

Symbolic Representations

Hope: Sets (transition relation, all reachable states) will have small formulas

We know

- + size of transition relation \cong size of circuit, software
- To represent a subset of $\{1, \dots, 2^k\}$ we need 2^k bits in general

We will try to find algorithms that tend to produce small formulas

Asynchronous Systems

skipped

Software

Modeling Software

Programs

Consist of

- consecution (;)
- `if`
- `while`
- `x := e`
- `skip`
- labels `L:`

Assume every line has a label.

Example

P::

```
l: cobegin P0 || P1 coend;
```

P0::

```
l0: while true do
```

```
    NC0: wait(turn = 0);
```

```
    CR0: turn := 1
```

```
end while
```

P1::

```
l1: while true do
```

```
    NC1: wait(turn = 1);
```

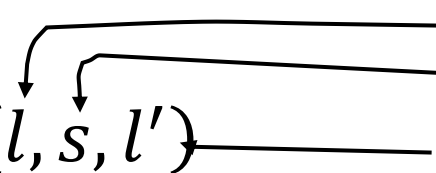
```
    CR1: turn := 0
```

```
end while
```

Translation

Define $same(Y) = \bigwedge_{y \in Y} y = Y'$

Define $\mathcal{C}(l, s, l')$



label of statement
statement
label of next statement

$$\mathcal{C}(l, v := e, l') =$$

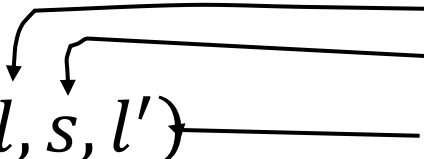
$$\mathcal{C}(l, skip, l') =$$

$$\mathcal{C}(l, (P; l' : P'), l'') =$$

Translation

Define $same(Y) = \bigwedge_{y \in Y} y = Y'$

Define $\mathcal{C}(l, s, l')$



label of statement
statement
label of next statement

$$\mathcal{C}(l, v := e, l') = pc = l \wedge pc' = l' \wedge v' = e \wedge same(V \setminus \{v\}),$$

$$\mathcal{C}(l, skip, l') = pc = l \wedge pc' = l' \wedge same(V),$$

$$\mathcal{C}(l, (P; l': P'), l'') = \mathcal{C}(l, P, l') \vee \mathcal{C}(l', P', l''),$$

Translation

$\mathcal{C}(l, \mathbf{if\ b\ then\ } l_1: P1 \mathbf{\ else\ } l_2: P2 \mathbf{\ end\ if, } l')$ =

Translation

$\mathcal{C}(l, \text{while } b \text{ do } l_1: P1 \text{ end while}, l') =$

Translation

$$\begin{aligned} \mathcal{C}(l, \mathbf{if\ } b \mathbf{\ then\ } l_1: P1 \mathbf{\ else\ } l_2: P2 \mathbf{\ end\ if}, l') = & \\ [pc = l \wedge b \wedge pc' = l_1 \wedge \mathit{same}(V)] \vee & \\ [pc = l \wedge \neg b \wedge pc' = l_2 \wedge \mathit{same}(V)] \vee & \\ \mathcal{C}(l_1, P1, l') \vee & \\ \mathcal{C}(l_2, P2, l') & \end{aligned}$$

$$\begin{aligned} \mathcal{C}(l, \mathbf{while\ } b \mathbf{\ do\ } l_1: P1; l_2: \mathbf{end\ while}, l') = & \\ [pc = l \wedge b \wedge pc' = l_1 \wedge \mathit{same}(V)] \vee & \\ [pc = l \wedge \neg b \wedge pc' = l' \wedge \mathit{same}(V)] \vee & \\ [pc = l_2 \wedge pc' = l \wedge \mathit{same}(V)] \vee & \\ \mathcal{C}(l_1, P1, l_2) & \end{aligned}$$

Concurrency

P :: cobegin

l1: P1 l1' ||

l2: P2 l2'

coend

Three program counters:

1. pc for the program that invokes cobegin
2. pc_1 for Thread 1
3. pc_2 for Thread 2

$pc = susp$ means that the program is not running.

$$\begin{aligned} \mathcal{C}(l, \mathbf{P}, l') &= (pc = l \wedge pc' = susp \wedge pc'_1 = l_1 \wedge pc'_2 = l_2 \wedge same(V)) \vee \\ & (pc = susp \wedge pc_1 = l'_1 \wedge pc_2 = l'_2 \wedge pc' = l' \wedge pc'_1 = susp \wedge pc'_2 = susp \wedge same(V)) \vee \\ & \bigvee_{i=1}^n (C(l_i, P_i, l'_i) \wedge same(V \setminus V_i) \wedge same(PC \setminus \{pc_i\})) \end{aligned}$$

Example

P0::

I0: **while** true **do**

 NC0: **wait**(turn = 0);

 CR0: turn := 1

end while

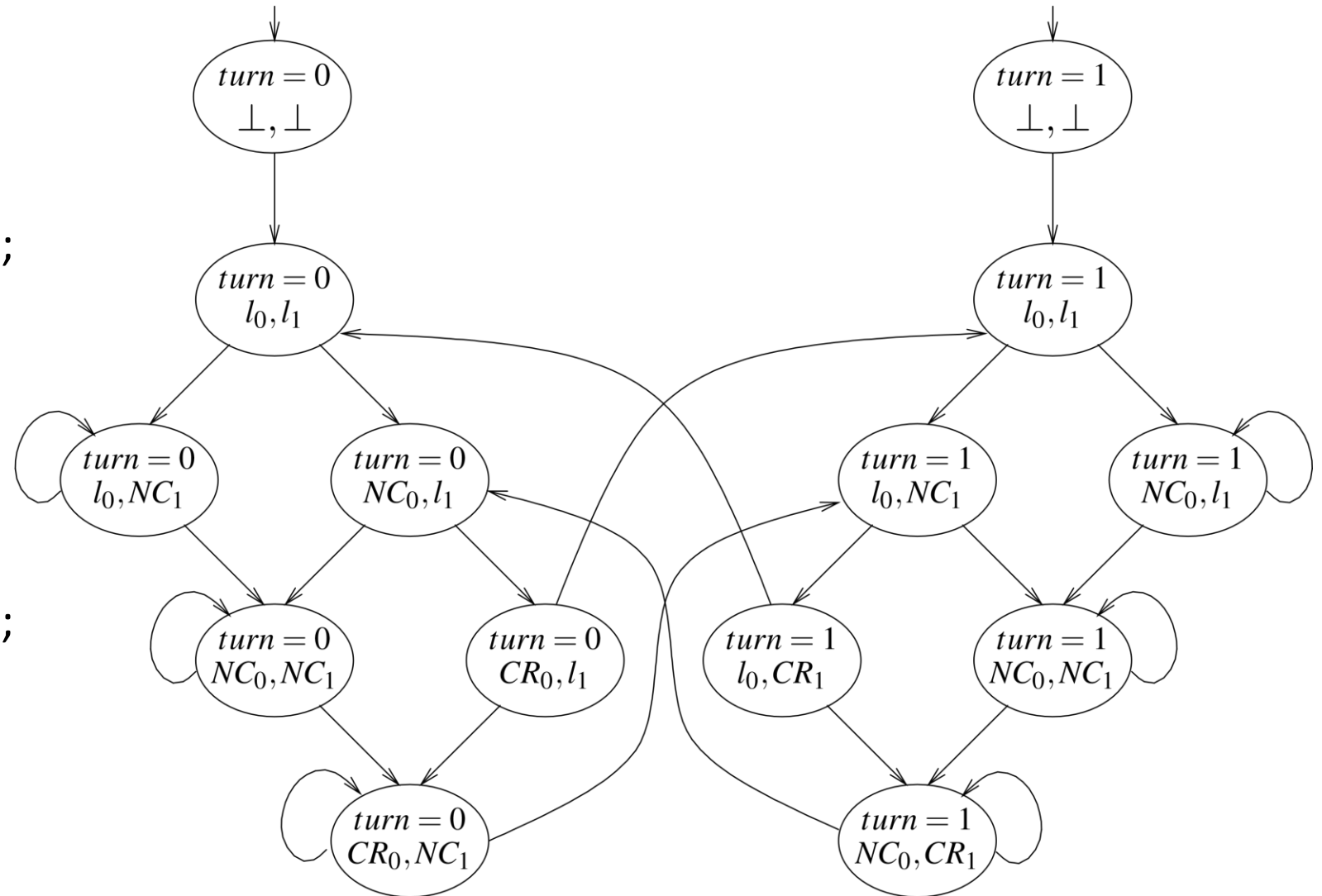
P1::

I1: **while** true **do**

 NC1: **wait**(turn = 1);

 CR1: turn := 0

end while



Termination

- Programs can end
- Kripke structures are not allowed to have dead ends, reminder:
 - $R \subseteq S \times S$ – left-total **transition relation**
 - For every $s \in S$ there exists $s' \in S$ such that $(s, s') \in R$
 - Left-total implies that every path is infinite
- We assume programs end in self loop that does nothing

Fairness

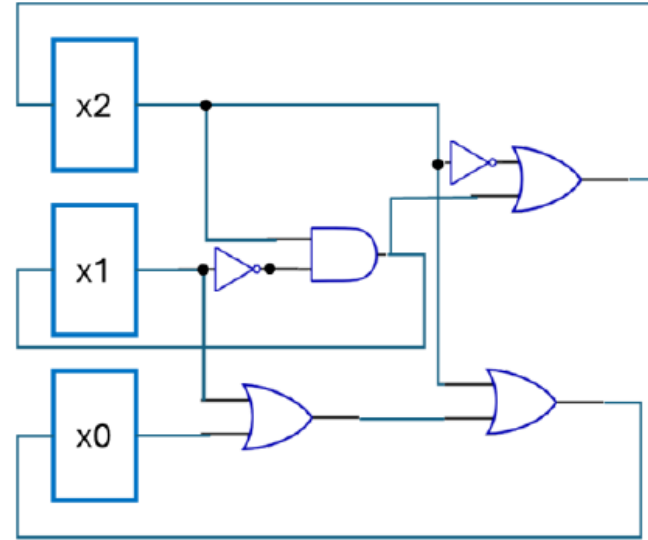
Fairness

- skipped

Consider the following synchronous circuit C . The initial value of the state variable x_0 is `true`. The initial values of x_1 and x_2 are unknown.

Model Checking (SS 2025) Homework 1

Deadline: **11 March 2025, 9:00 am**
Submit your solution through TeachCenter



Task 1. [30 points] State the formula S_0 that represents the set of initial states and the formula R that represents the transition relation of C .

Task 2. [30 points] Draw the Kripke structure $M = (S, S_0, R, AP, L)$ that represents C . (Add an incoming arrow for each initial state.)

Task 3. [40 points] We want to use BMC to check whether x_0 is always `true`.

3.1 Will BMC find a counterexample? If so, what is the smallest k such that BMC finds a counterexample. [10 points]

3.2 Write the BMC formula for $k = 2$. (You can use S_0 and R in your formula. [15 points]

3.3 Is the formula satisfiable? Explain. [15 points]