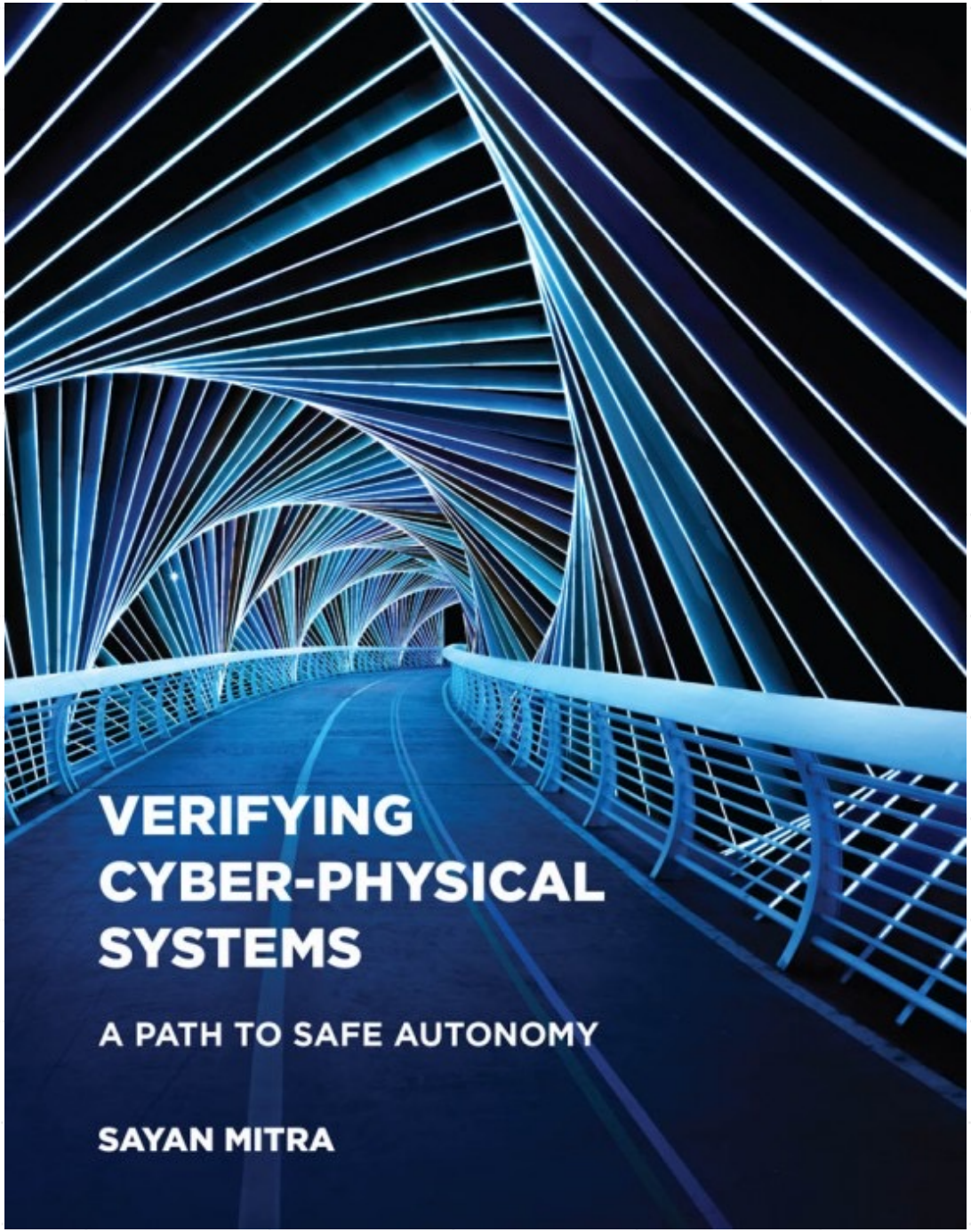
Verifying Cyber-Physical Systems

Chapter 1 – 2

Abenezer Taye

Feb 09, 2022

Chapter 1: INTRODUCTION

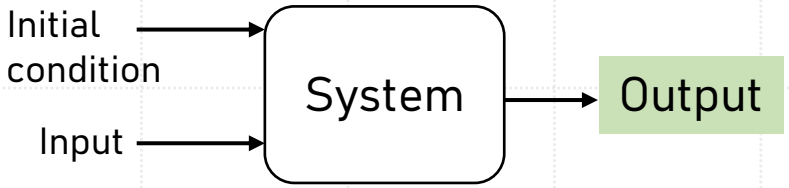


Introduction

- **Cyber Physical System (CPS):** systems that are controlled by computer
- **Requirements for CPS:** the desired or claimed behaviors of the system (aka specifications or properties)

Testing Vs Verification

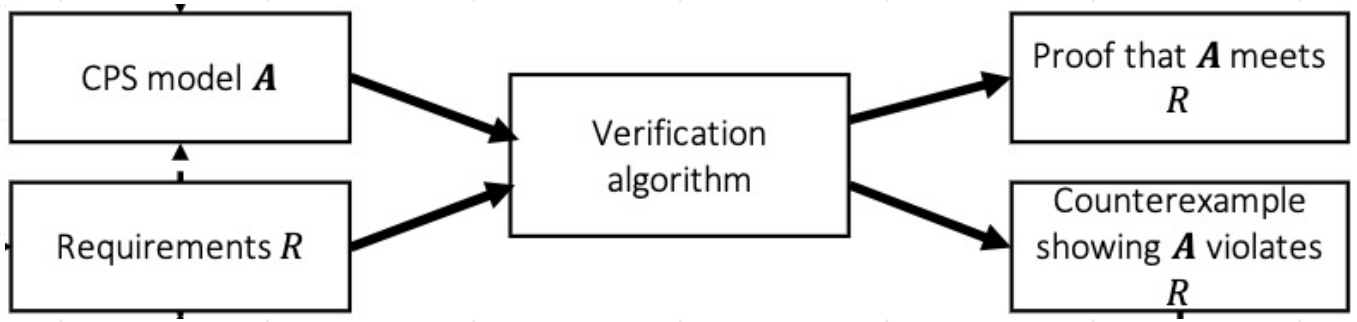
- **Testing**: analyzing the resulting behavior data to check whether the requirement is met



- For many systems, using testing for establishing system requirements is impossible
  - The amount of data required to guarantee a probability of 10^{-9} fatality/hr is 10^9 hrs of data $\approx 30\times10^9$miles (Shalev-Shwartz et al. (2017))
- **Verification**: mathematically proves that a model of the system meets the requirements

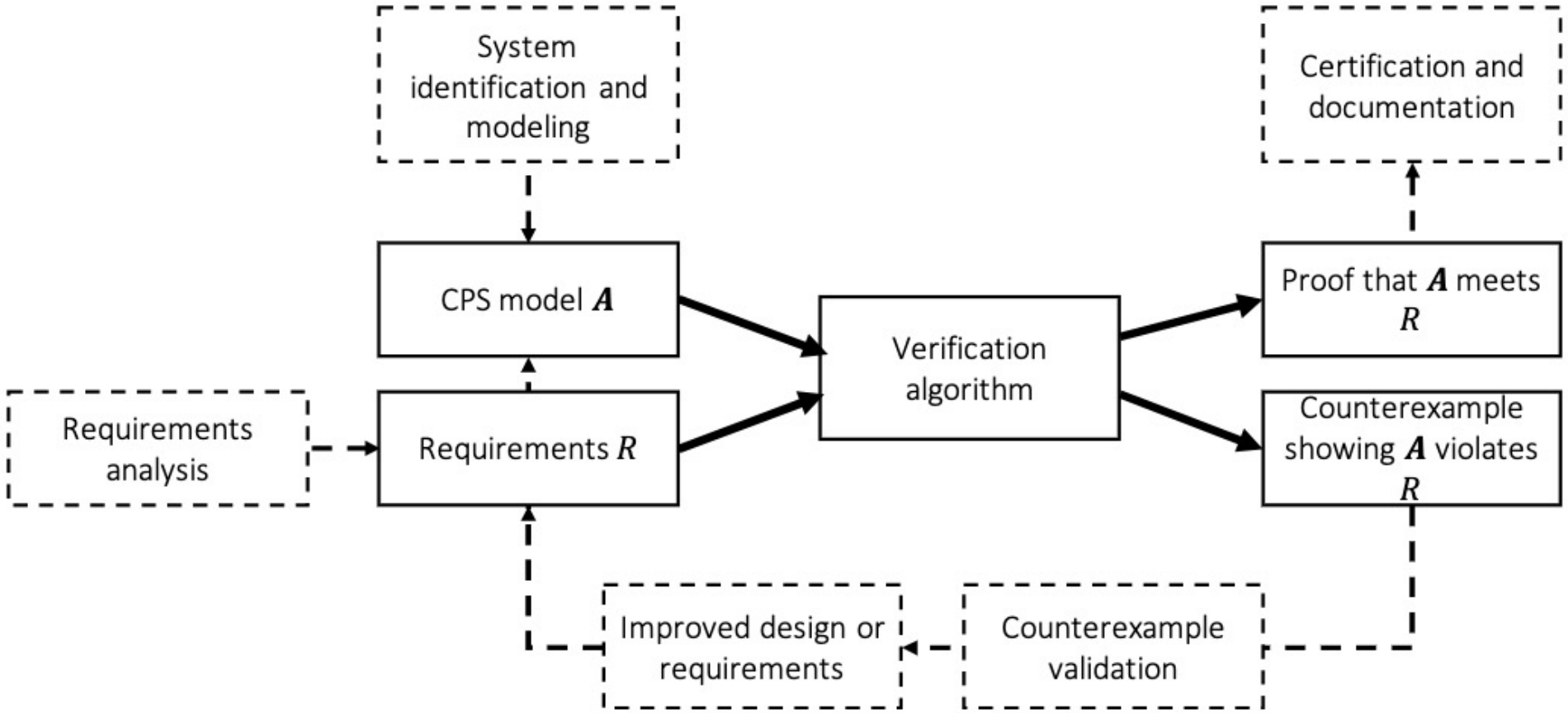
Verification

- The verification process looks like:



- CPS model (A): is usually a hybrid model (ODE + automata)
- There could be two possible meanings to counterexample α:
  - **Spurious counterexample**: because of the gap between model and the actual system
    - Solution ➔ Model validation
  - **Real counterexample**:
    - A design bug ➔ debug and modify the system
    - The requirements are wrong/inconsistent ➔ modify the requirements

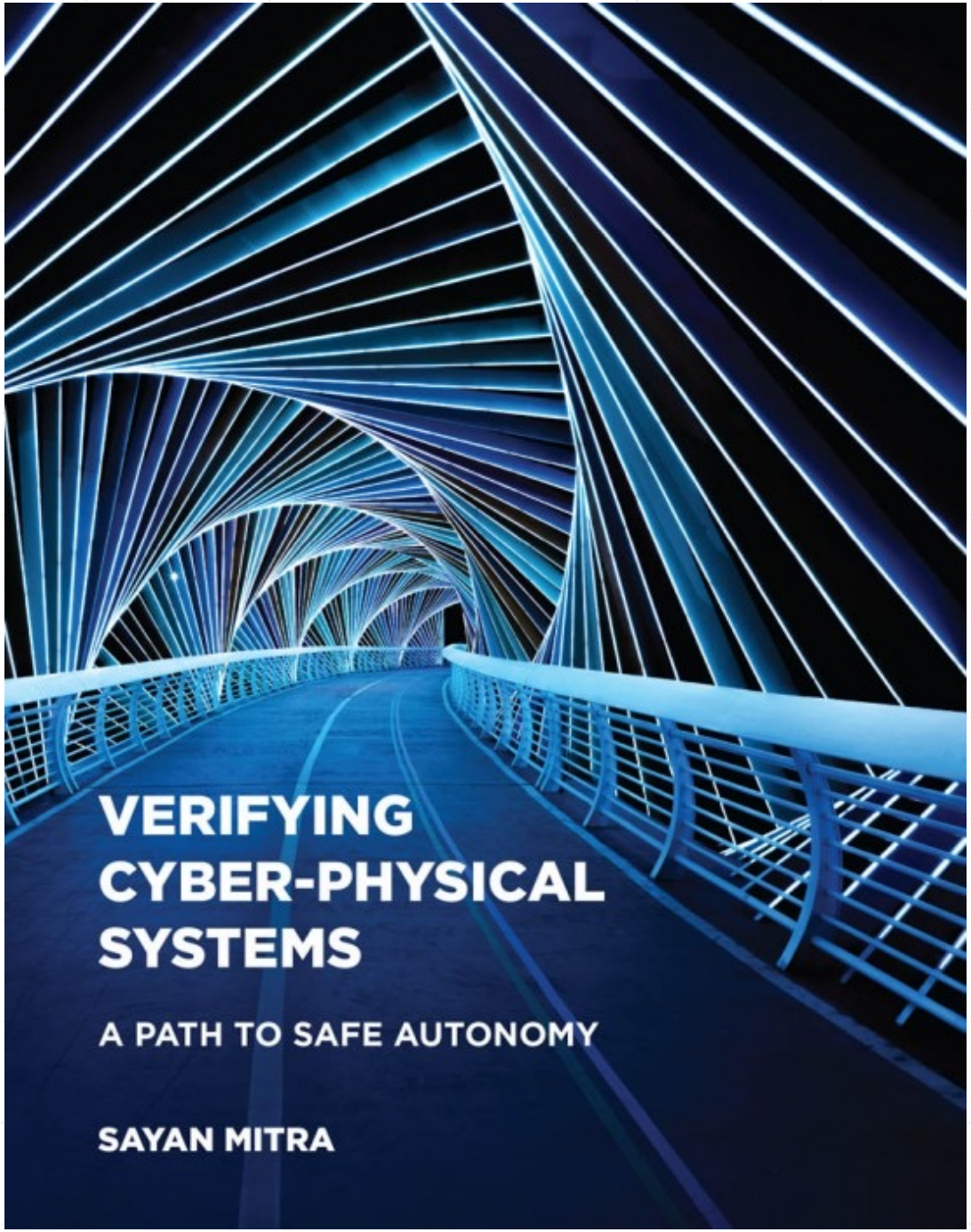
System Design Ecosystem



Challenges of Verification

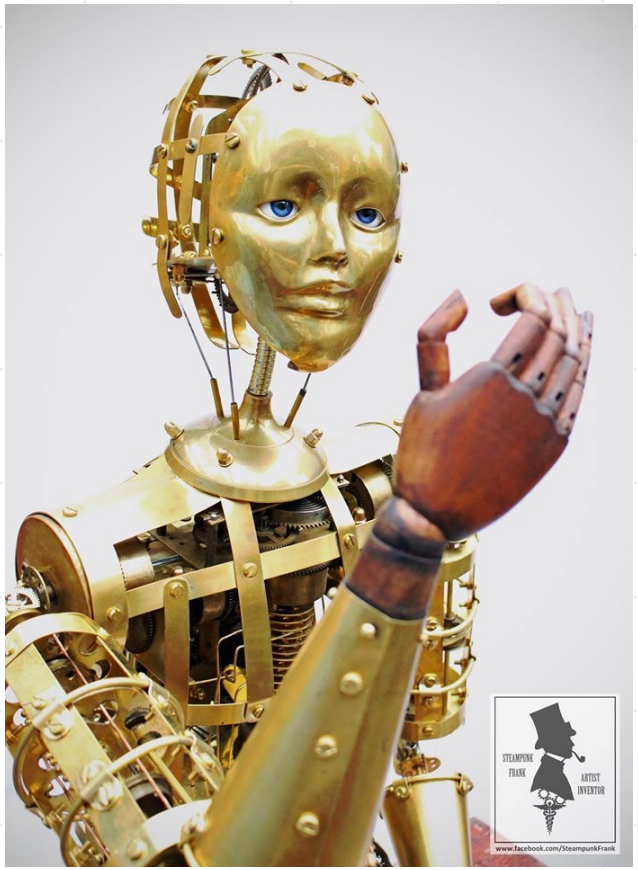
1. **Theoretical undecidability :**
   - There is no algorithm that checks whether a model of CPS meets a requirement.
   - To address this → relaxing the verification question (e.g. bounded time horizon or tolerate some false positives)
2. **Practical Scalability :**
   - Lack of scalability is a huge challenge for the progress in verification
3. **Missing Models :**
   - Complete models of CPS systems are difficult to come by
     - Oftentimes models lack succinct descriptions, are distributed, and are protected intellectual property
4. **Learning-enabled systems and missing requirements :**
   - Recent CPS systems use DNN ➔ lack models and requirements

Chapter 2: MODELING COMPUTATION

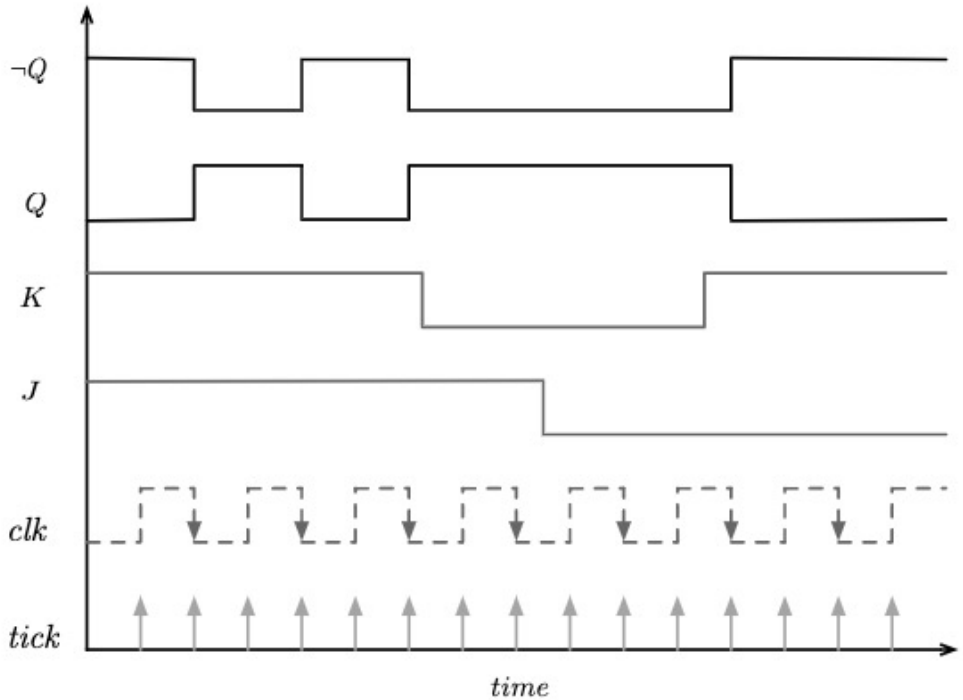


Introduction

- **Aim:** how can we develop a mathematical model for computation
- We use differential equations to model physical systems ➔ how about computation?
- **Automaton**: 19th century word used to describe mechanical dolls
- In mathematics, we use automaton to describe the rules of state change of any computational process
  - Aka, **discrete-time transition system** or **a state machine**



Example: JK Flip-Flop



clk	J	k	Q
0	0	0	previous
0	0	1	0
0	1	0	1
0	1	1	toggle

```
1  automaton FlipFlop
       actions
3          tick

5      variables
           J : Bool := 1; K : Bool := 1
7          clk : Bool := 0
           Q : Bool := 0

transitions                                          10
   tick
   pre true                                          12
   eff clk := ¬clk
      if clk = 0 then                                14
         if J = 1 ∧ K = 1 then Q := ¬Q
         elseif J = 0 ∧ K = 1 then Q := 0            16
         elseif J = 1 ∧ K = 0 then Q := 1 fi
      fi                                             18
```

Figure 2.2
A specification of a JK flip-flop.

Parts of specification

1. **Actions**: lists various types of rules for state change
2. **Variables**: lists the state variables
3. **Transitions**: lists the state change rules for each of the actions

Specifying automata

1. **State variables:**
   - A state represents the information necessary for predicting the future of an automaton
2. **Valuation:**
   - Maps each state variable name to a value $\mathbf{v} = (J \mapsto 1, K \mapsto 1, clk \mapsto 0, Q \mapsto 0).$
3. **Predicates:**
   - A Boolean–valued formula involving the variables in V (we use it as a requirement)
     - For example: the state $\mathbf{v} = (J \mapsto 1, K \mapsto 1, clk \mapsto 0, Q \mapsto 0).$ satisfies the formula $\phi_1 := (J = 0 \ \vee \ K = 1)$
     - Because $(\mathbf{v} \lceil J = 0 \ \vee \ \mathbf{v} \lceil K = 1) \equiv (0 = 0 \ \vee \ 1 = 1) \equiv true.$
4. **Transitions:**
   - Specify the rules for state change in a computation

Automata

- An automaton $\mathcal{A}$ is a tuple $(\boldsymbol{V}, \boldsymbol{\Theta}, \boldsymbol{A}, \boldsymbol{\mathcal{D}})$, where

1. V is a set of state variables and val(V) is valuations of V
2. $\Theta \subseteq val(C)$ is a set of start states
3. A is a set of actions
4. $\mathcal{D} \subseteq val(V) \times A \times val(V)$ is called the set of transitions

$$\mathbf{v}' \lceil J = \mathbf{v} \lceil J, \mathbf{v}' \lceil K = \mathbf{v} \lceil K, \mathbf{v}' \lceil clk = \neg \mathbf{v} \lceil clk$$
$$\neg \mathbf{v}' \lceil clk \ \wedge \mathbf{v} \lceil J \ \wedge \neg \mathbf{v} \lceil K \Rightarrow \mathbf{v}' \lceil Q,$$
$$\neg \mathbf{v}' \lceil clk \ \wedge \neg \mathbf{v} \lceil J \ \wedge \mathbf{v} \lceil K \Rightarrow \neg \mathbf{v}' \lceil Q,$$
$$\neg \mathbf{v}' \lceil clk \ \wedge \mathbf{v} \lceil J \ \wedge \mathbf{v} \lceil K \Rightarrow (\mathbf{v}' \lceil Q = \neg \mathbf{v}' \lceil Q).$$

```
transitions                                              10
  tick
  pre true                                               12
  eff clk := ¬clk
    if clk = 0 then                                      14
      if J = 1 ∧ K = 1 then Q := ¬Q
      elseif J = 0 ∧ K = 1 then Q := 0                   16
      elseif J = 1 ∧ K = 0 then Q := 1 fi
    fi                                                   18
```

Semantics

- **Execution** $(Exces_{\mathcal{A}}(\Theta))$: an alternating, possibly infinite, sequence of states and actions $v_0, a_1, v_2, \ldots$
  1. Each v_i is a state val(V)
  2. Each a_i is an action in A
  3. Each consecutive triple (v_i, a_{i+1}, v_{i+1}) is a valid transition in $\mathcal{D}$
- **Reachable state** $(Reach_{\mathcal{A}}(\Theta))$: a state is reachable if there exists a finite execution that ends at v
- **Invariant:** a set of states $\boldsymbol{S}$ that contains all of $\mathcal{A}$'s reachable states

Thank You!!!