State Diagrams

Part c: Transition Functions and Counting States

Ian Ludden

Learning Objectives

By the end of this lesson, you will be able to:

Learning Objectives

By the end of this lesson, you will be able to:

- Formally define a transition function.

Learning Objectives

By the end of this lesson, you will be able to:

- Formally define a transition function.
- Evaluate ways of storing functions in a computer.

Learning Objectives

By the end of this lesson, you will be able to:

- Formally define a transition function.
- Evaluate ways of storing functions in a computer.
- Compute the number of states for an example system.

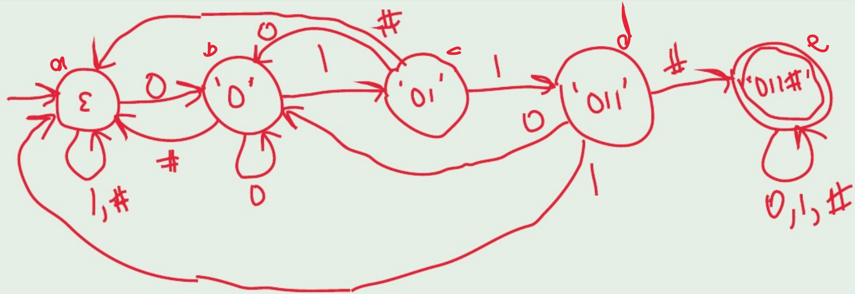
Transition Functions

Consider a state diagram with a set of states S, a start state $s_0 \in S$, end state(s) $Q \subseteq S$, a set of actions A, and a transition function δ. Formally, δ has type signature $\delta : S \times A \to \mathbb{P}(S)$.

Transition Functions

Consider a state diagram with a set of states S, a start state $s_0 \in S$, end state(s) $Q \subseteq S$, a set of actions A, and a transition function δ. Formally, δ has type signature $\delta : S \times A \to \mathbb{P}(S)$.

Example: Garage Door Keypad



$\delta(b, 1) = \{c\}$ $\delta(e, \cdot) = \{e\}$

$\delta(a, 0) = \{b\}$

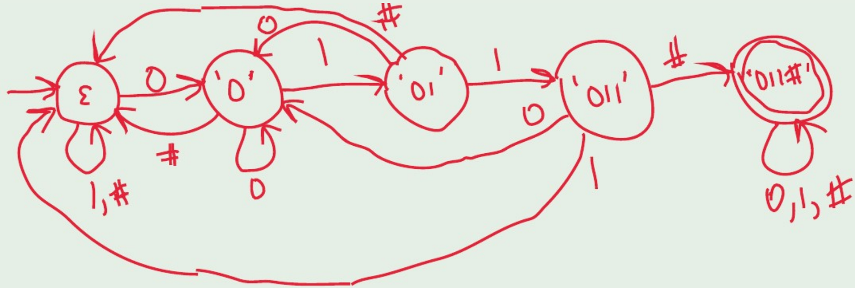
Transition Functions

Consider a state diagram with a set of states S, a start state $s_0 \in S$, end state(s) $Q \subseteq S$, a set of actions A, and a transition function δ. Formally, δ has type signature $\delta : S \times A \to \mathbb{P}(S)$.

Transition Functions

Consider a state diagram with a set of states S, a start state $s_0 \in S$, end state(s) $Q \subseteq S$, a set of actions A, and a transition function δ. Formally, δ has type signature $\delta : S \times A \to \mathbb{P}(S)$.

Example 1: Garage Door Keypad



Why Output a Set of States?

$\delta : S \times A \to \mathbb{P}(S)$

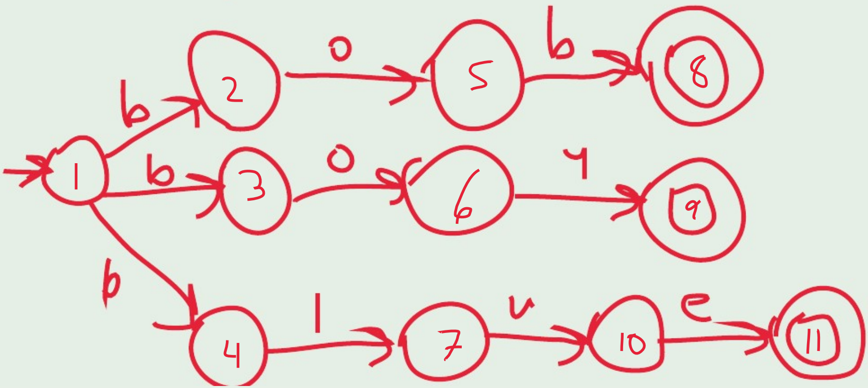
Why Output a Set of States?

$\delta : S \times A \to \mathbb{P}(S)$

$\delta(1, b) = \{2, 3, 4\}$ $\delta(6, u) = \emptyset$

Example 2: Phone Lattice

{boy, bob, blue}



Storing Transition Functions

Storing Transition Functions

- Option 1: List of input/output pairs

$$[((1,b),2), ((1,b),3), ((5,y),10), \ldots]$$

Storing Transition Functions

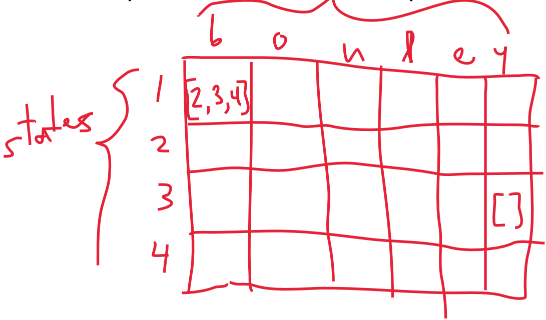
- Option 1: List of input/output pairs
  - Pro: Easy to implement

Storing Transition Functions

- Option 1: List of input/output pairs
  - Pro: Easy to implement
  - Con: Slow to look up

Storing Transition Functions

- Option 1: List of input/output pairs
  - Pro: Easy to implement
  - Con: Slow to look up
- Option 2: 2D array of state/action pairs with lists of output states



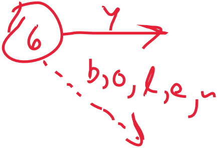
Storing Transition Functions

- Option 1: List of input/output pairs
  - Pro: Easy to implement
  - Con: Slow to look up
- Option 2: 2D array of state/action pairs with lists of output states
  - Pro: Easy to implement

Storing Transition Functions

- Option 1: List of input/output pairs
  - Pro: Easy to implement
  - Con: Slow to look up
- Option 2: 2D array of state/action pairs with lists of output states
  - Pro: Easy to implement
  - Con: Lots of wasted space (many state diagrams are sparse)

few edges



Storing Transition Functions

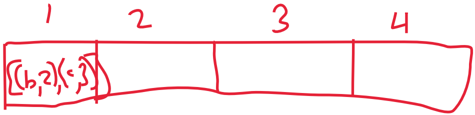
- Option 1: List of input/output pairs
  - Pro: Easy to implement
  - Con: Slow to look up
- Option 2: 2D array of state/action pairs with lists of output states
  - Pro: Easy to implement
  - Con: Lots of wasted space (many state diagrams are sparse)
- Option 3: 1D array of states with lists of possible actions and next states

Storing Transition Functions

- Option 1: List of input/output pairs
  - Pro: Easy to implement
  - Con: Slow to look up
- Option 2: 2D array of state/action pairs with lists of output states
  - Pro: Easy to implement
  - Con: Lots of wasted space (many state diagrams are sparse)
- Option 3: 1D array of states with lists of possible actions and next states
  - Pro: More compact

Storing Transition Functions

- Option 1: List of input/output pairs
  - Pro: Easy to implement
  - Con: Slow to look up
- Option 2: 2D array of state/action pairs with lists of output states
  - Pro: Easy to implement
  - Con: Lots of wasted space (many state diagrams are sparse)
- Option 3: 1D array of states with lists of possible actions and next states
  - Pro: More compact
  - Con: More bookkeeping

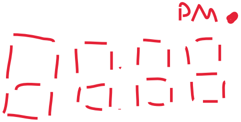


Storing Transition Functions

- Option 1: List of input/output pairs
  - Pro: Easy to implement
  - Con: Slow to look up
- Option 2: 2D array of state/action pairs with lists of output states
  - Pro: Easy to implement
  - Con: Lots of wasted space (many state diagrams are sparse)
- Option 3: 1D array of states with lists of possible actions and next states
  - Pro: More compact
  - Con: More bookkeeping
- Option 4: Get fancy with hash functions (hash tables, dictionaries, etc.)

Counting States: Exact

A simple digital clock has four digits (HH:MM) and an LED indicator for “p.m.” How many states does the clock have:



Counting States: Exact

A simple digital clock has four digits (HH:MM) and an LED indicator for "p.m." How many states does the clock have:

- under normal operation?

Counting States: Exact

12 · 60 · 2

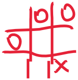
A simple digital clock has four digits (HH:MM) and an LED indicator for "p.m." How many states does the clock have:

- under normal operation? 1,440 (one for each minute)
- if the digits are not restricted to valid times?

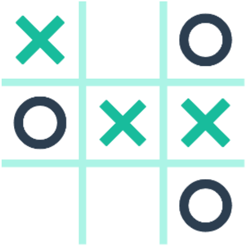
HH:MM p.m.

$10 \cdot 10 \cdot 10 \cdot 10 \cdot 2 = 20{,}000$

Counting States: Estimate



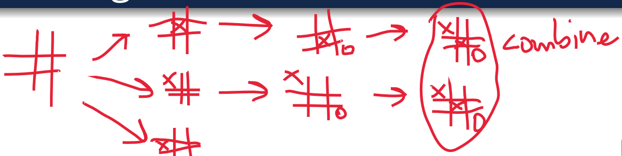
- Tic-Tac-Toe (image source)



$\leq 3^9$ possible states

$\approx \log_2 3^9 \approx 9 \cdot \log_2 3$

Counting States: Estimate



- Tic-Tac-Toe (image source)
- Connect 4 (image source)

$\leq 3^{42}$ states

- Tic-Tac-Toe (image source)
- Connect 4 (image source)
- Chess (image source)

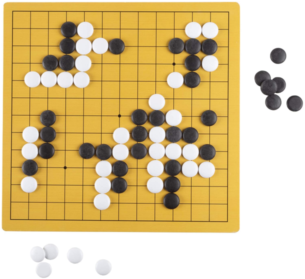
$$\frac{64!}{32! \cdot (8!)^2 \cdot (2!)^6} \approx 10^{43}$$

→ Claude Shannon

Counting States: Estimate

- Tic-Tac-Toe (image source)
- Connect 4 (image source)
- Chess (image source)
- Go (image source)

$\leq 3^{361}$ states



Counting States: Estimate

- Tic-Tac-Toe (image source)
- Connect 4 (image source)
- Chess (image source)
- Go (image source)
- Starcraft II... (relevant tweet)

10^{1000}

Counting States: Estimate

- Tic-Tac-Toe (image source)
- Connect 4 (image source)
- Chess (image source)
- Go (image source)
- Starcraft II... (relevant tweet)

Fun fact: There are over 43 quintillion (4.3×10^{19}) permutations of the 3×3 Rubik's Cube. (Link to source)

Recap: Learning Objectives

By the end of this lesson, you will be able to:

- Formally define a transition function.
- Evaluate ways of storing functions in a computer.
- Compute the number of states for an example system.