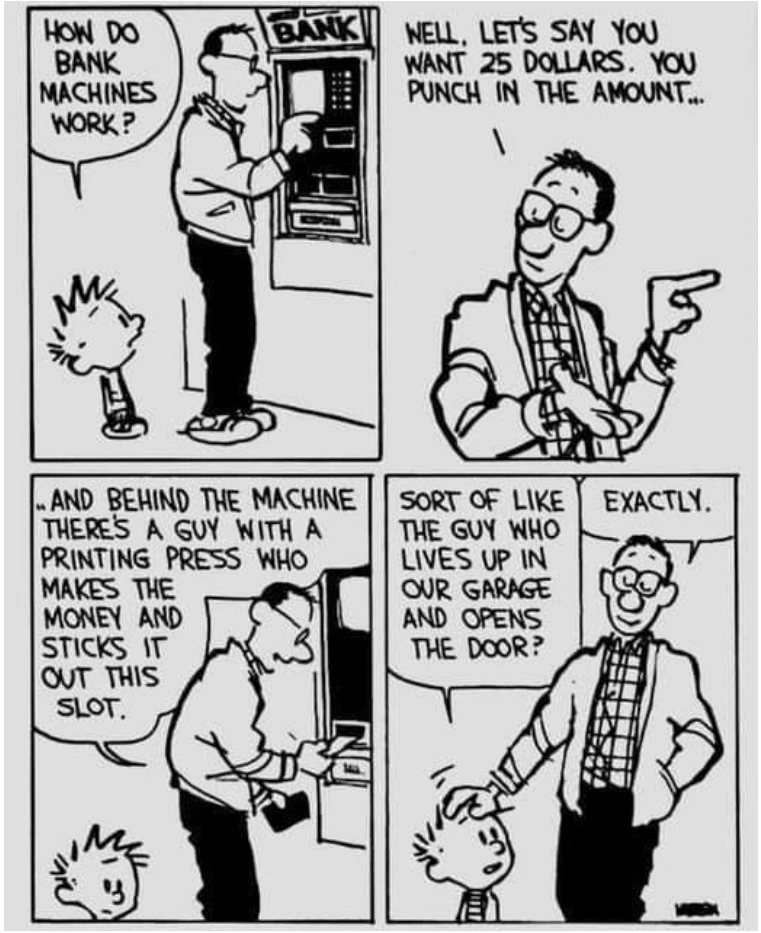


Deterministic Finite Automata (DFA)

CSE 311: Foundations of Computing I
Lecture 21

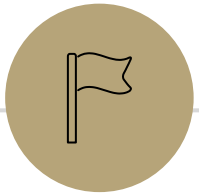
Announcements

- HW6 due tonight at 11:59pm on Gradescope.
- Midterm Corrections due tonight at 11:59pm on Gradescope.
  - No Late Days permitted
- HW7 published tonight (Last HW!)
  - Everything is auto-graded, using a CSE online tool called Grin

    (a demo video of how to use Grin will be linked in the HW writeup)
  - No Late Days permitted

Review

- A **language** is a set of strings.
  - **Regular Languages**: Languages that can be specified by a Regular Expression (RegEx).
  - **Context Free Languages**: Languages that can be generated by a Context Free Grammar (CFG).
- A computer is said to **recognize** a language if it can distinguish which strings are in a language vs. which are not.
  - **Deterministic Finite Automata** (DFA): One model of computation.
  - **Nondeterministic Finite Automata** (NFA): Another model of computation.

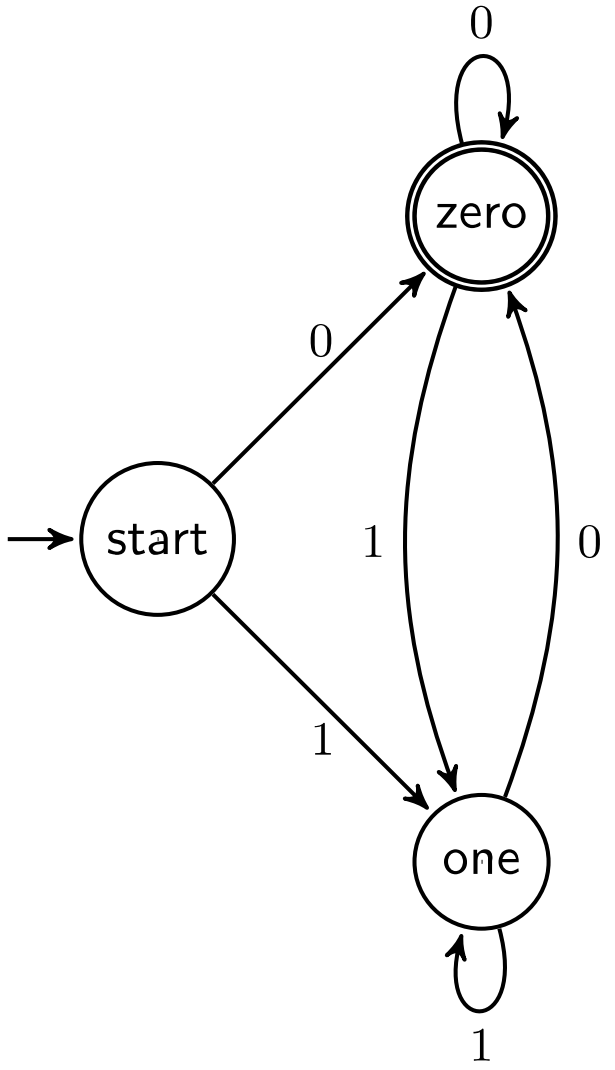


Deterministic Finite Automata (DFA)

Deterministic Finite Automata (DFA)

What are the components of a DFA?

- Finite number of states
  - Accept States, indicated by double or bolded borders
  - Reject States, indicated by regular borders
  - 1 special state called the “start state”, indicated by the incoming arrow. The start state might be an accept or a reject state.
- Labelled directed edges between states
  - From each state, there is **exactly one** outgoing edge per character in the alphabet (e.g. 2 total outgoing edges for each state if the alphabet is {0, 1}).
  - Edges can go from a state to itself. Those are called “self-loops”.
  - If two or more edges are going from state A to state B, we just draw one edge with two labels.



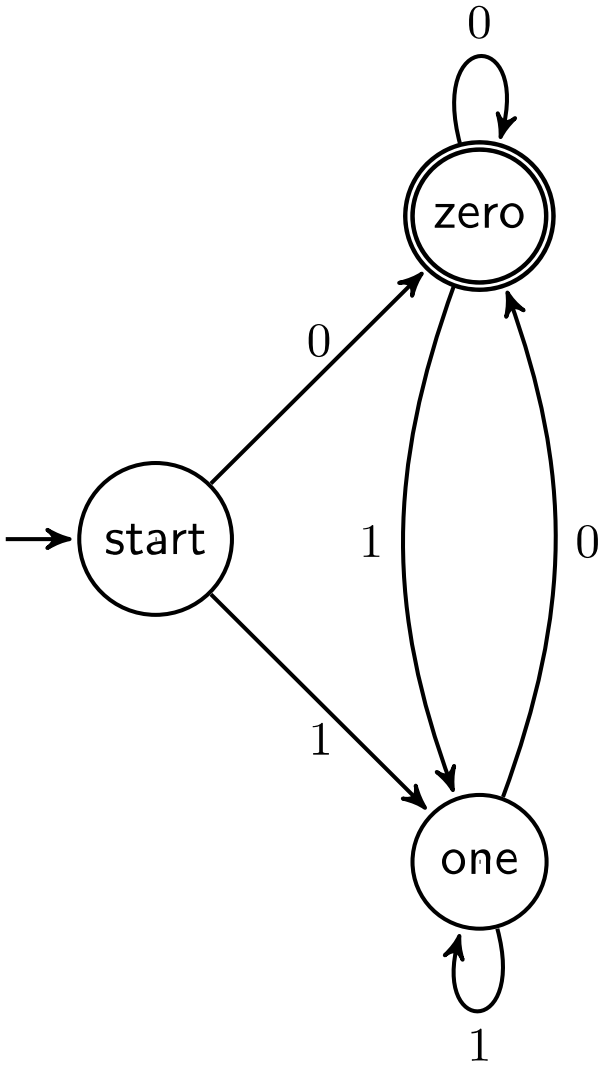
Deterministic Finite Automata (DFA)

How does a DFA processes strings?

- The DFA takes a string as input.
- Starting at the start state, the DFA reads one character at a time.
- When it reads the character it follows the arrow labeled with that character to its next state.
- After reading the last character, the DFA accepts the string if and only if the processing ends in an accept state.

For Example:

1 0 1 1



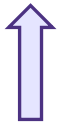
Deterministic Finite Automata (DFA)

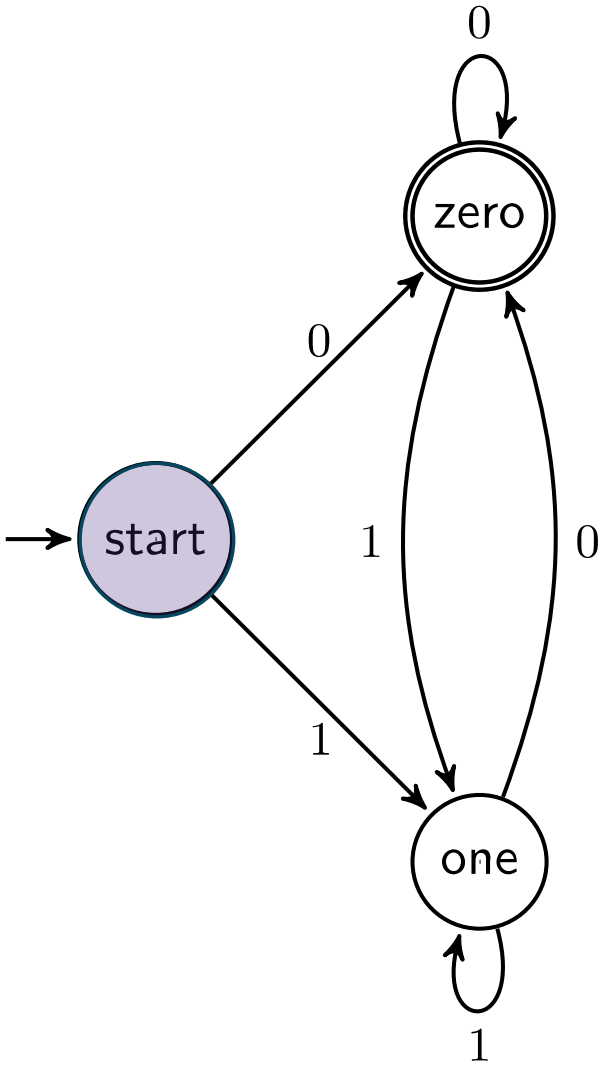
How does a DFA processes strings?

- The DFA takes a string as input.
- Starting at the start state, the DFA reads one character at a time.
- When it reads the character it follows the arrow labeled with that character to its next state.
- After reading the last character, the DFA accepts the string if and only if the processing ends in an accept state.

For Example:

1 0 1 1





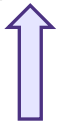
Deterministic Finite Automata (DFA)

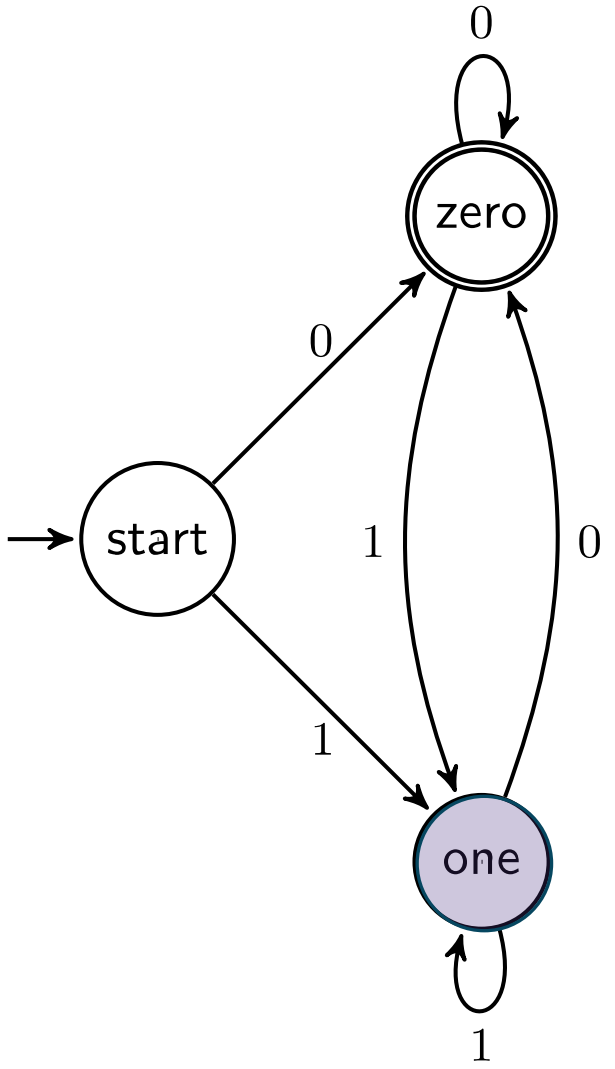
How does a DFA processes strings?

- The DFA takes a string as input.
- Starting at the start state, the DFA reads one character at a time.
- When it reads the character it follows the arrow labeled with that character to its next state.
- After reading the last character, the DFA accepts the string if and only if the processing ends in an accept state.

For Example:

1 0 1 1





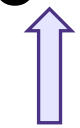
Deterministic Finite Automata (DFA)

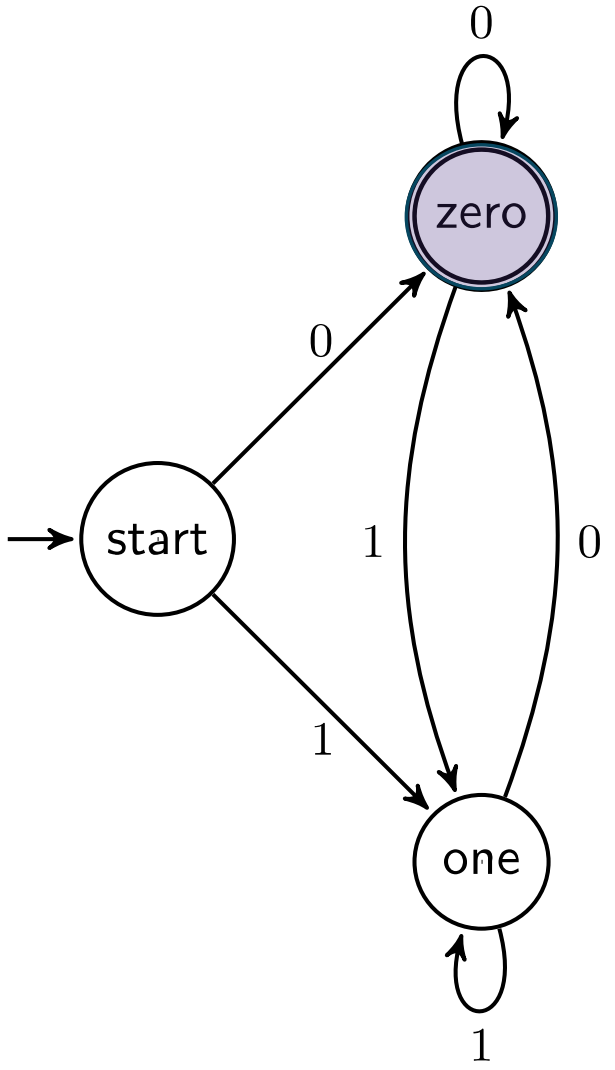
How does a DFA processes strings?

- The DFA takes a string as input.
- Starting at the start state, the DFA reads one character at a time.
- When it reads the character it follows the arrow labeled with that character to its next state.
- After reading the last character, the DFA accepts the string if and only if the processing ends in an accept state.

For Example:

1 0 1 1





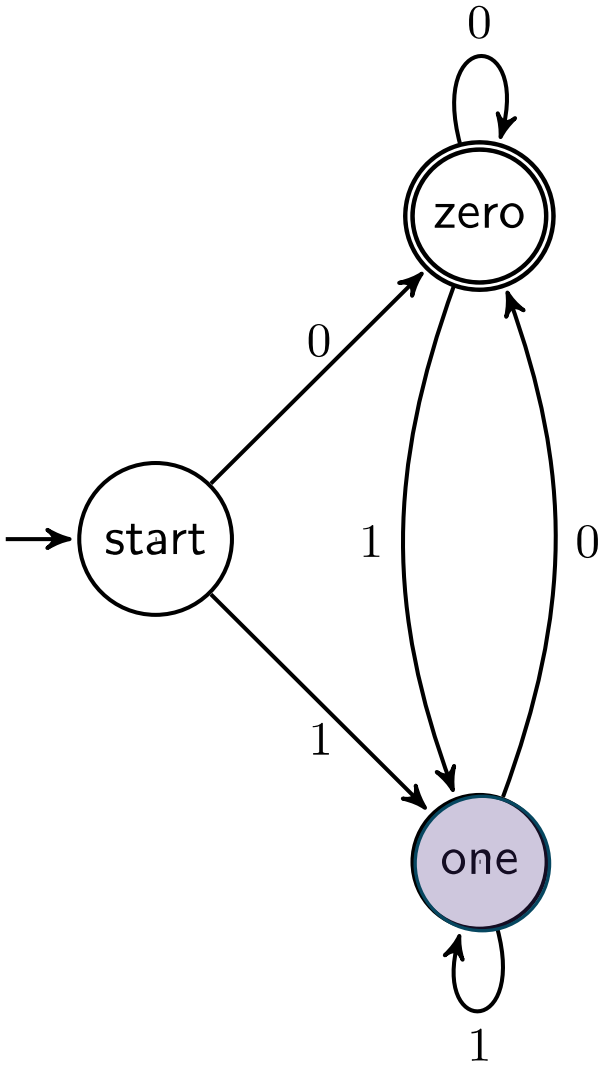
Deterministic Finite Automata (DFA)

How does a DFA processes strings?

- The DFA takes a string as input.
- Starting at the start state, the DFA reads one character at a time.
- When it reads the character it follows the arrow labeled with that character to its next state.
- After reading the last character, the DFA accepts the string if and only if the processing ends in an accept state.

For Example:

1 0 1 1



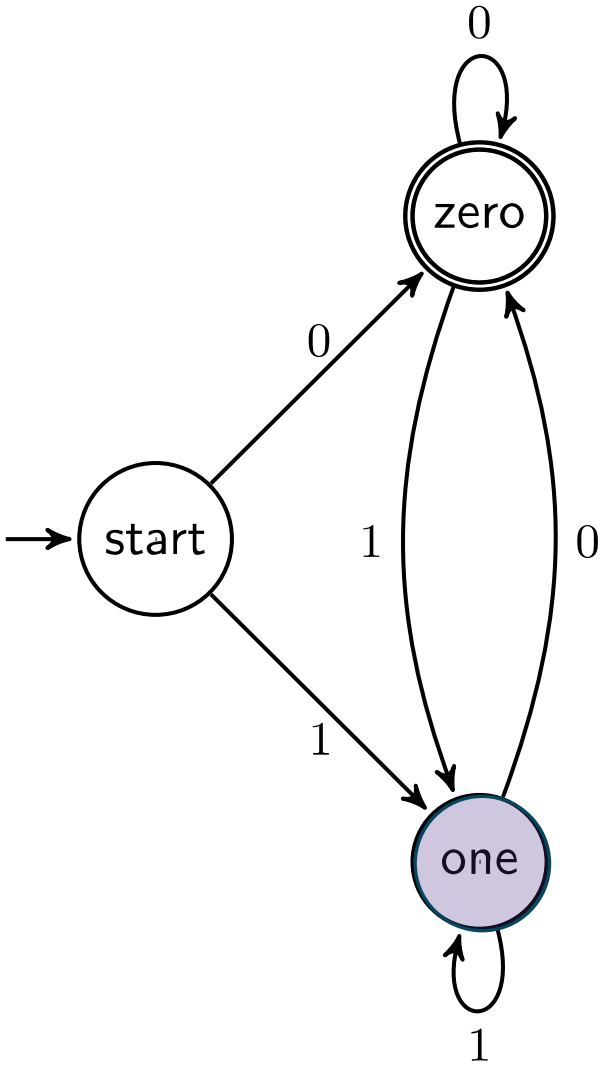
Deterministic Finite Automata (DFA)

How does a DFA processes strings?

- The DFA takes a string as input.
- Starting at the start state, the DFA reads one character at a time.
- When it reads the character it follows the arrow labeled with that character to its next state.
- After reading the last character, the DFA accepts the string if and only if the processing ends in an accept state.

For Example:

1 0 1 1



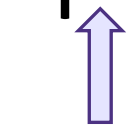
Deterministic Finite Automata (DFA)

How does a DFA processes strings?

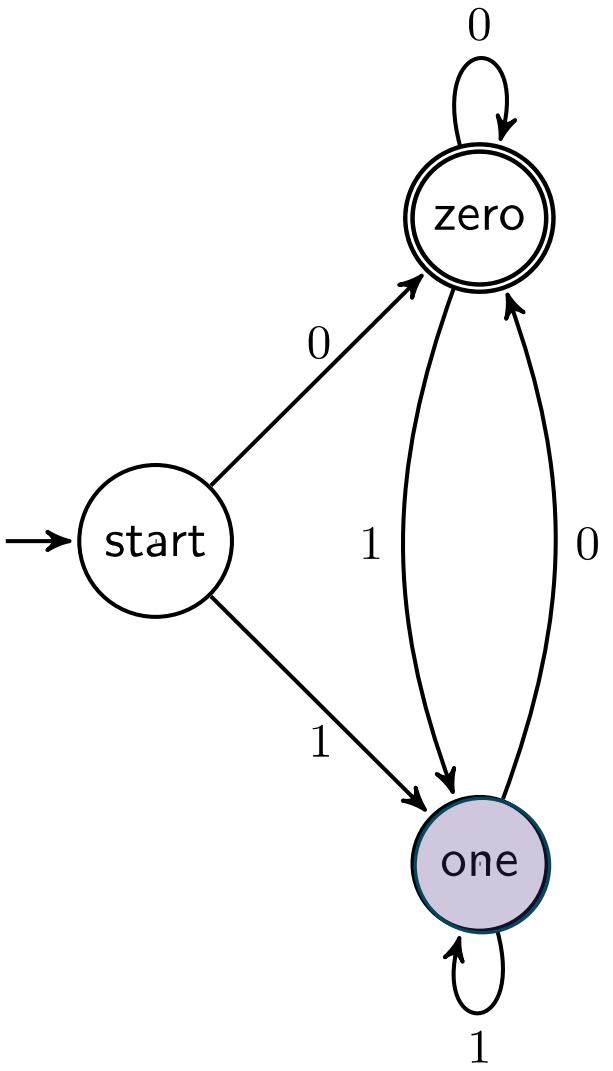
- The DFA takes a string as input.
- Starting at the start state, the DFA reads one character at a time.
- When it reads the character it follows the arrow labeled with that character to its next state.
- After reading the last character, the DFA accepts the string if and only if the processing ends in an accept state.

For Example:

1 0 1 1



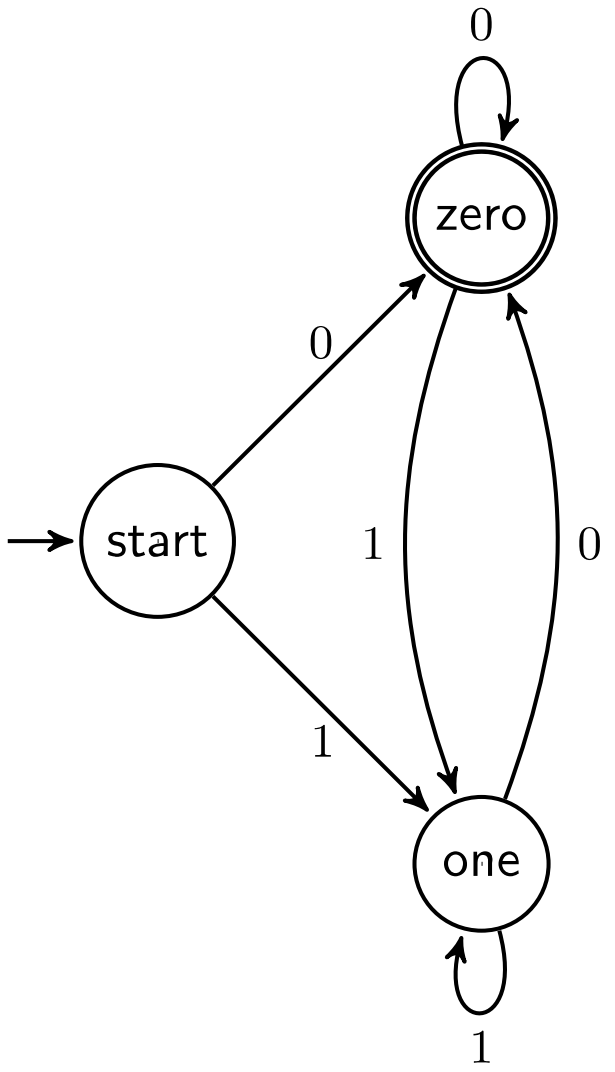
Rejected!



Analyzing the Language Recognized by a DFA

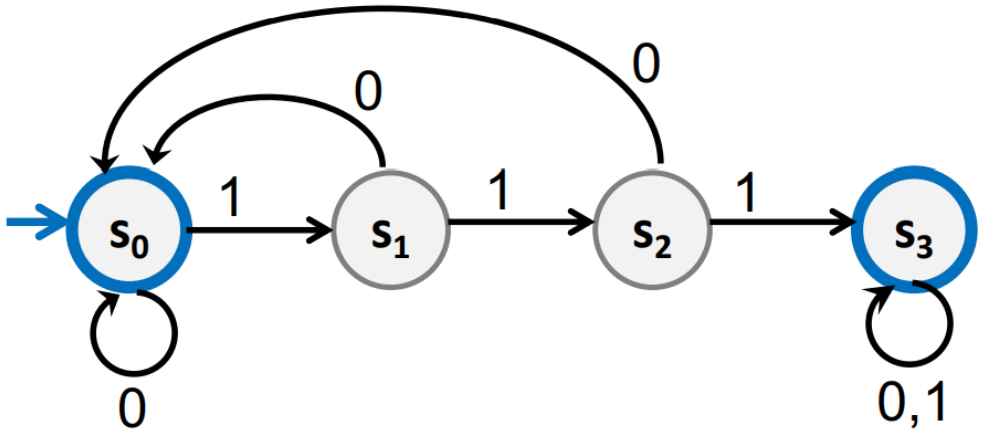
In general, what language does a DFA recognize?

In this example, what language does this DFA recognize?



Example

What language does this DFA recognize?



Deterministic Finite Automata (DFA)

Now we know how to analyze a DFA and recognize what language it accepts. But can we start from a language, and *build* the corresponding DFA?

1. Determine which states you'll need, and what each state represents.
   - For each state, identify it's an accept or reject state.
   - Determine which state is the start state, by analyzing which state ε should end in.
2. Determine how to connect each state with labelled edges
   - May need to adjust your set of states when doing this
   - Check that each state has exactly one outgoing edge per character in the alphabet.
3. Check that your DFA is correct by testing several strings that should be accepted, and several that should be rejected.

Example

1. Determine which states, including accept or reject & start state.
2. Determine how to connect each state with labelled edges. Each state has one outgoing per character in the alphabet.
3. Check that the DFA is correct by testing strings.

Design a DFA that accepts the language of strings over $\{0, 1, 2\}$ where the sum of digits $\% \, 3$ is 0.

Exercise

1. Determine which states, including accept or reject & start state.
2. Determine how to connect each state with labelled edges. Each state has one outgoing per character in the alphabet.
3. Check that the DFA is correct by testing strings.

DFA for the set of binary strings with a $\mathbf{1}$ in the 3rd position from the start.

E.g. $\mathbf{0010}$ should be accepted, $\mathbf{01000}$ and $\mathbf{11}$ should be rejected.

What's with the name?

Deterministic Finite Automata

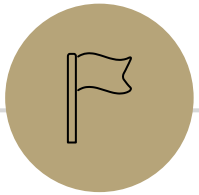
Deterministic:
Given the same input, will always follow predictable steps to produce the same output.

If a DFA processes the same string, we can follow the same series of steps to determine if the string is accepted or rejected.

Finite: Not infinite; eventually ends.

A DFA has a finite number of states.

Automata: A machine that performs a function according to predetermined instructions.



DFAs in the Real World

DFAs in the Real World

DFAs are not as powerful machines as our modern computers. However, in many contexts they're more than enough.

For example, DFAs are often used to implement…

Elevators

Vending Machines

Traffic Lights

DFAs in the Real World

Let's see how DFAs could be used to implement a simple Vending Machine.

Suppose the Vending Machine accepts $5¢$ and $10¢$ coins, and dispenses a soda automatically when it reaches a $25¢$ balance.

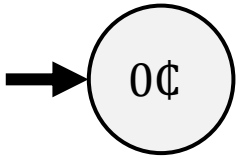
We wish to design a DFA that keeps track of the current balance in the machine.

DFAs in the Real World

Let's see how DFAs could be used to implement a simple Vending Machine.

Suppose the Vending Machine accepts $5¢$ and $10¢$ coins, and dispenses a soda automatically when it reaches a $25¢$ balance.

We wish to design a DFA that keeps track of the current balance in the machine.

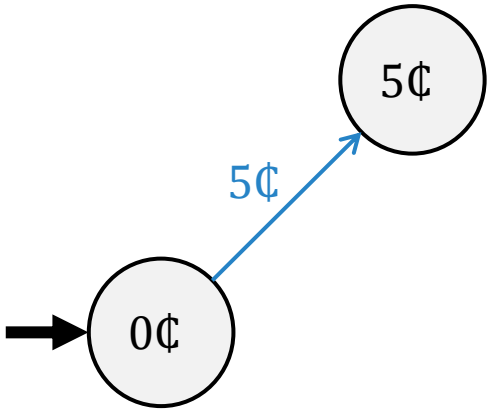


DFAs in the Real World

Let's see how DFAs could be used to implement a simple Vending Machine.

Suppose the Vending Machine accepts $5¢$ and $10¢$ coins, and dispenses a soda automatically when it reaches a $25¢$ balance.

We wish to design a DFA that keeps track of the current balance in the machine.

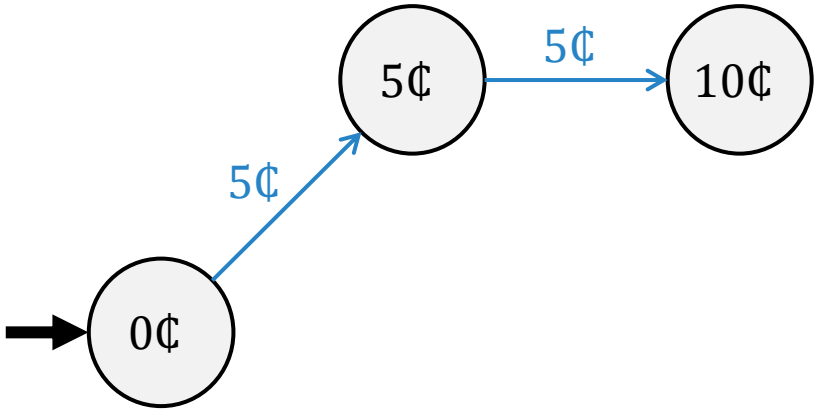


DFAs in the Real World

Let's see how DFAs could be used to implement a simple Vending Machine.

Suppose the Vending Machine accepts $5¢$ and $10¢$ coins, and dispenses a soda automatically when it reaches a $25¢$ balance.

We wish to design a DFA that keeps track of the current balance in the machine.

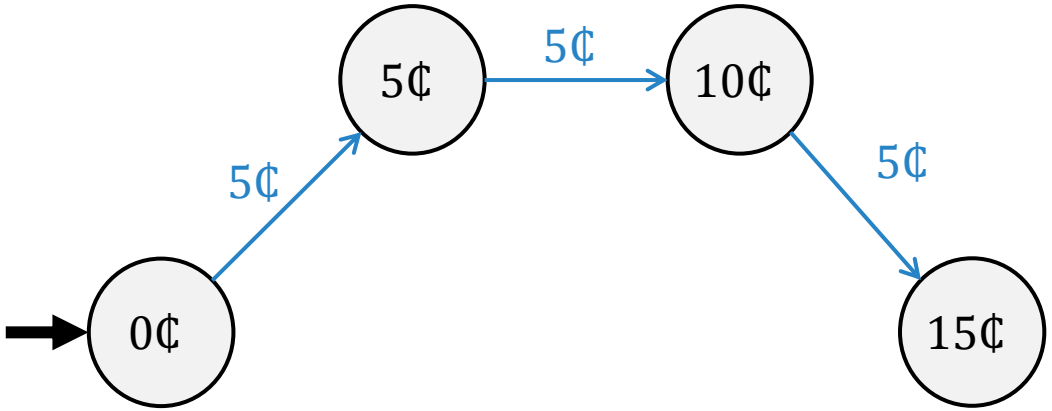


DFAs in the Real World

Let's see how DFAs could be used to implement a simple Vending Machine.

Suppose the Vending Machine accepts $5¢$ and $10¢$ coins, and dispenses a soda automatically when it reaches a $25¢$ balance.

We wish to design a DFA that keeps track of the current balance in the machine.

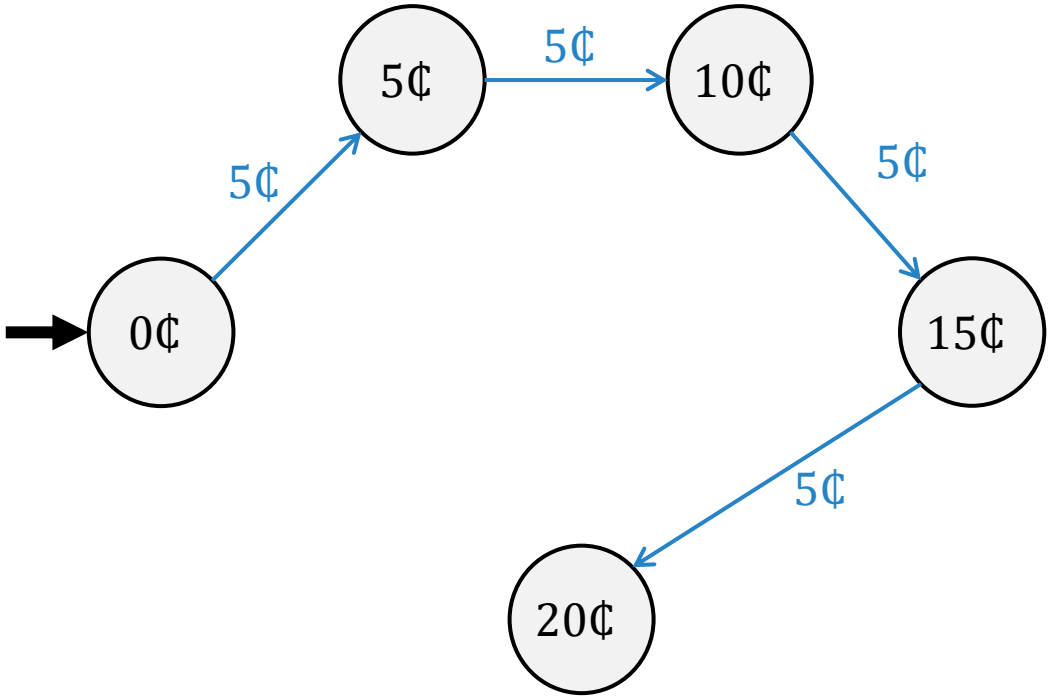


DFAs in the Real World

Let's see how DFAs could be used to implement a simple Vending Machine.

Suppose the Vending Machine accepts $5¢$ and $10¢$ coins, and dispenses a soda automatically when it reaches a $25¢$ balance.

We wish to design a DFA that keeps track of the current balance in the machine.

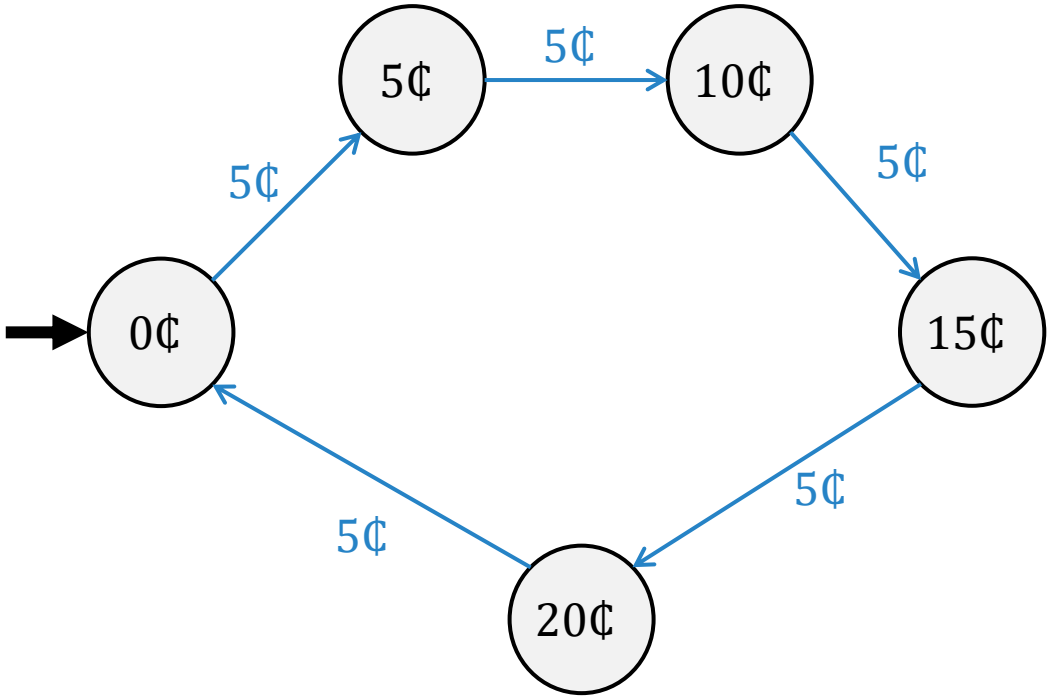


DFAs in the Real World

Let's see how DFAs could be used to implement a simple Vending Machine.

Suppose the Vending Machine accepts $5¢$ and $10¢$ coins, and dispenses a soda automatically when it reaches a $25¢$ balance.

We wish to design a DFA that keeps track of the current balance in the machine.

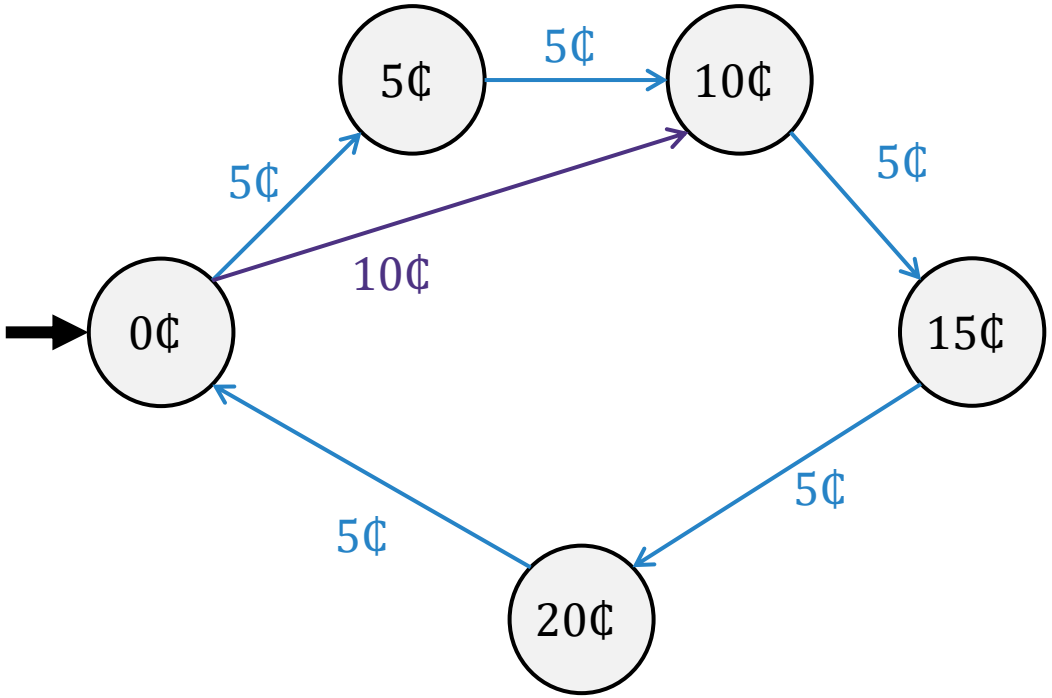


DFAs in the Real World

Let's see how DFAs could be used to implement a simple Vending Machine.

Suppose the Vending Machine accepts $5¢$ and $10¢$ coins, and dispenses a soda automatically when it reaches a $25¢$ balance.

We wish to design a DFA that keeps track of the current balance in the machine.

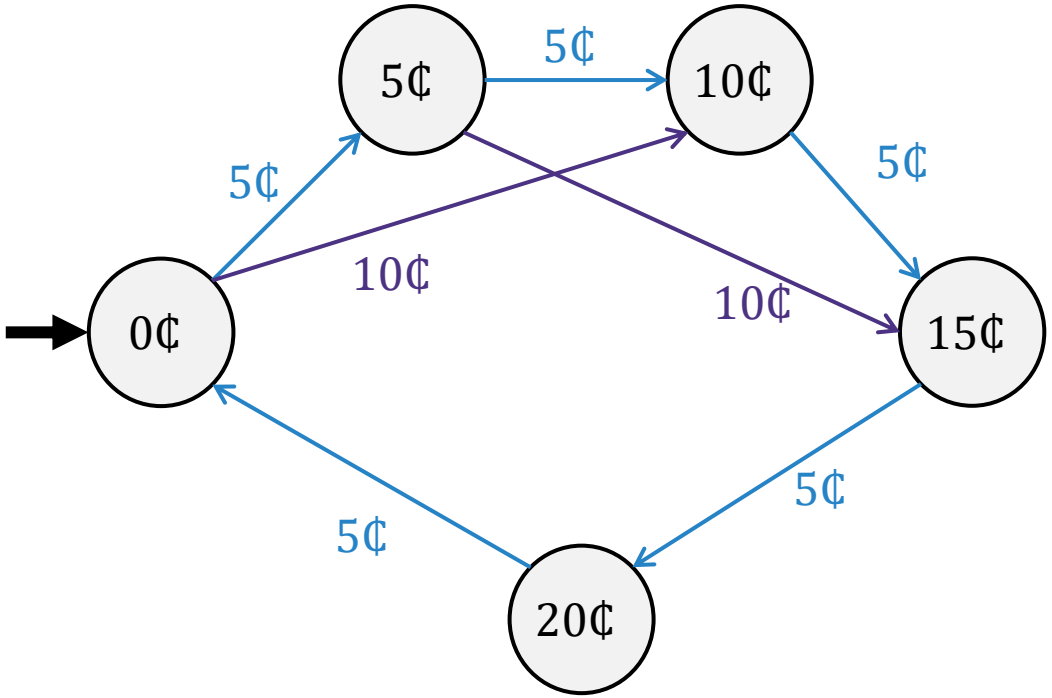


DFAs in the Real World

Let's see how DFAs could be used to implement a simple Vending Machine.

Suppose the Vending Machine accepts $5¢$ and $10¢$ coins, and dispenses a soda automatically when it reaches a $25¢$ balance.

We wish to design a DFA that keeps track of the current balance in the machine.

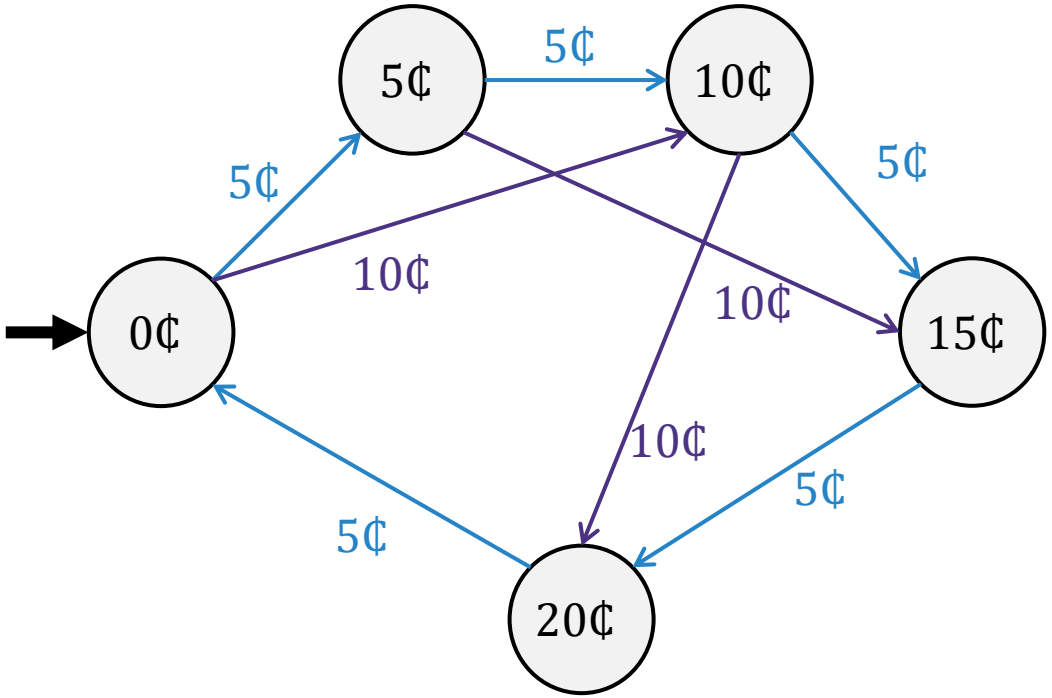


DFAs in the Real World

Let's see how DFAs could be used to implement a simple Vending Machine.

Suppose the Vending Machine accepts $5¢$ and $10¢$ coins, and dispenses a soda automatically when it reaches a $25¢$ balance.

We wish to design a DFA that keeps track of the current balance in the machine.

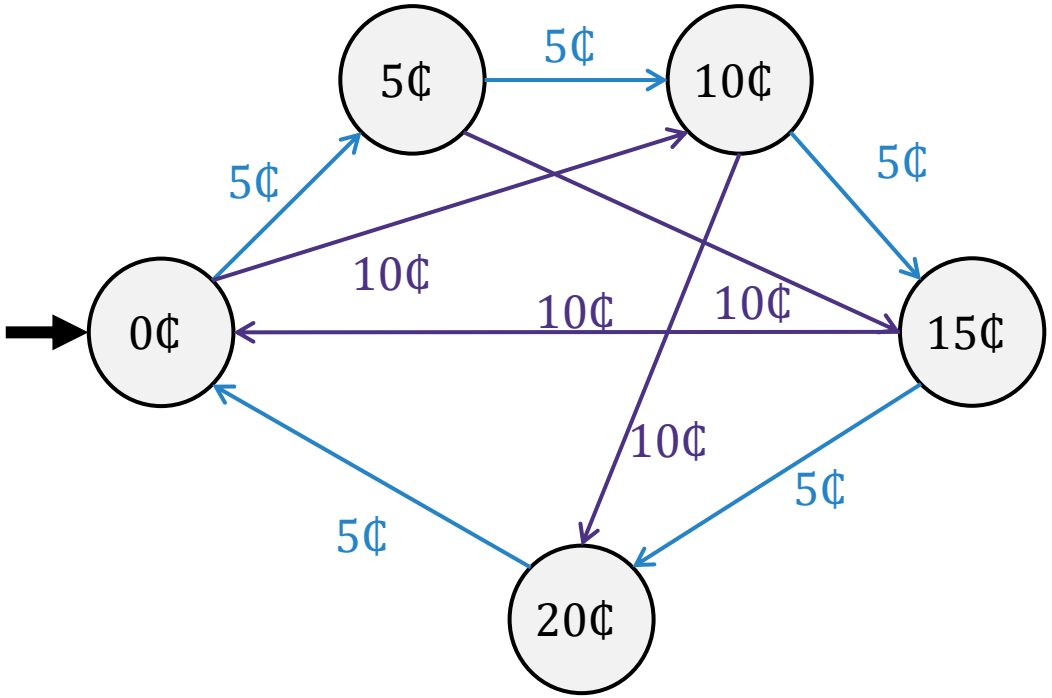


DFAs in the Real World

Let's see how DFAs could be used to implement a simple Vending Machine.

Suppose the Vending Machine accepts $5¢$ and $10¢$ coins, and dispenses a soda automatically when it reaches a $25¢$ balance.

We wish to design a DFA that keeps track of the current balance in the machine.

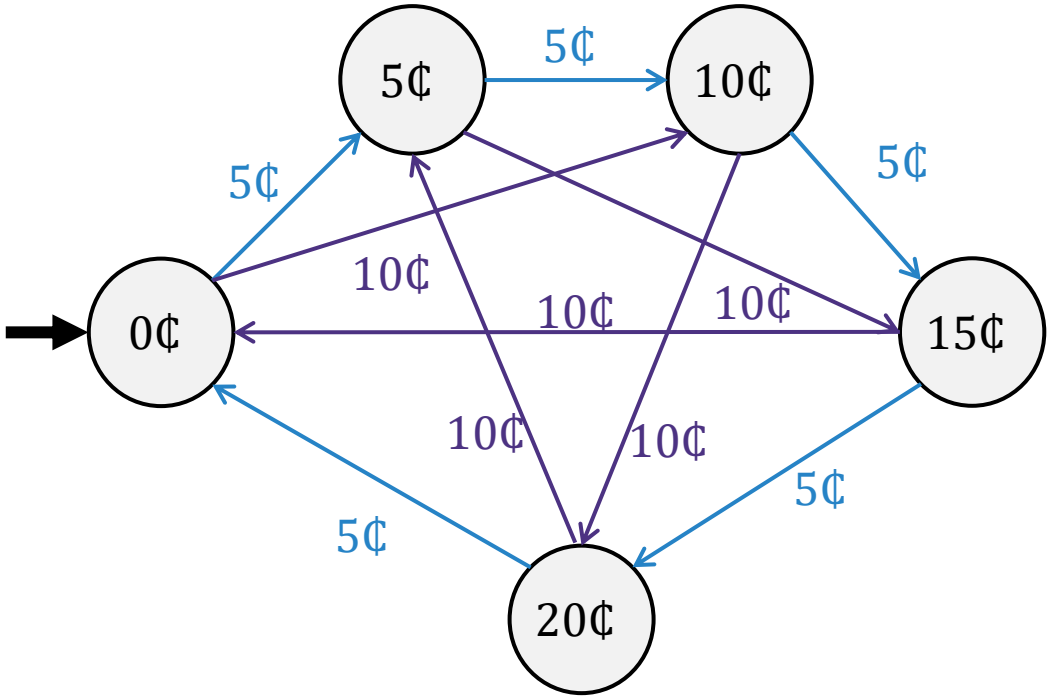


DFAs in the Real World

Let's see how DFAs could be used to implement a simple Vending Machine.

Suppose the Vending Machine accepts $5¢$ and $10¢$ coins, and dispenses a soda automatically when it reaches a $25¢$ balance.

We wish to design a DFA that keeps track of the current balance in the machine.

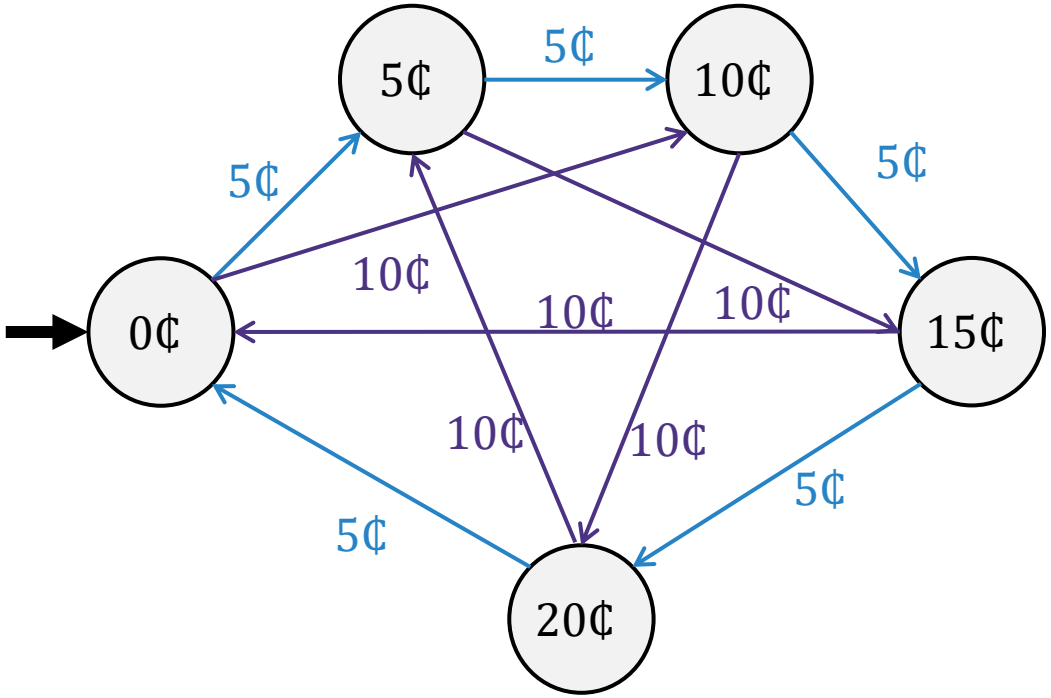


DFAs in the Real World

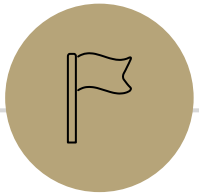
Let's see how DFAs could be used to implement a simple Vending Machine.

Suppose the Vending Machine accepts $5¢$ and $10¢$ coins, and dispenses a soda automatically when it reaches a $25¢$ balance.

We wish to design a DFA that keeps track of the current balance in the machine.



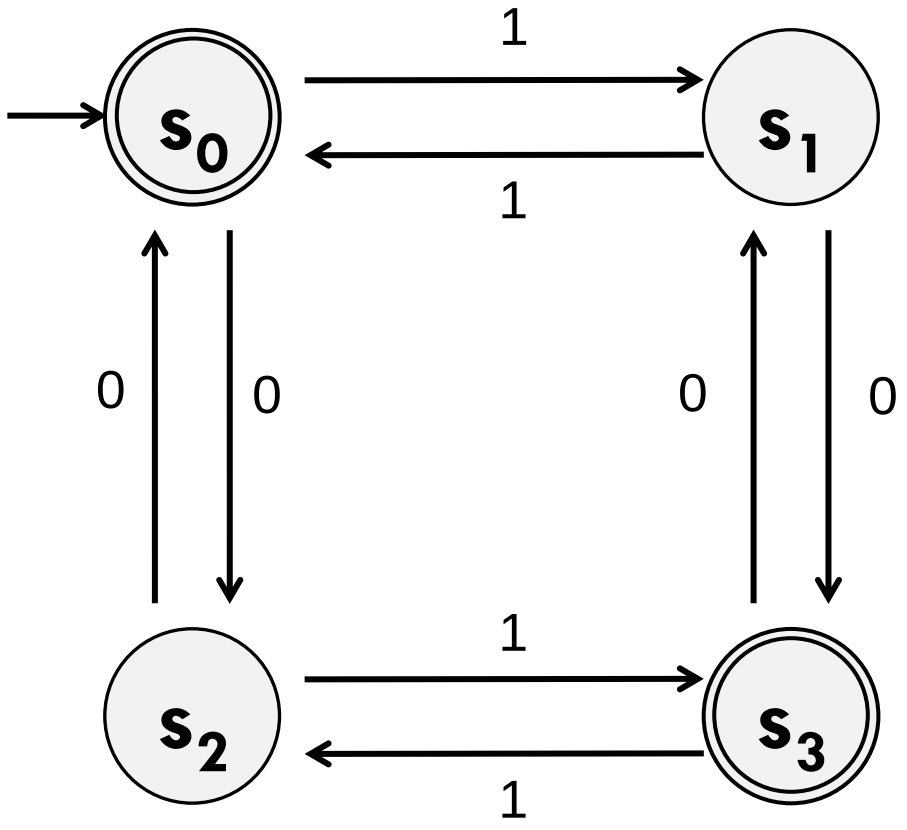
Note: This application is slightly different than before, since there are no "accept" states, and the DFA's purpose is not to accept a particular language.



DFA Minimization

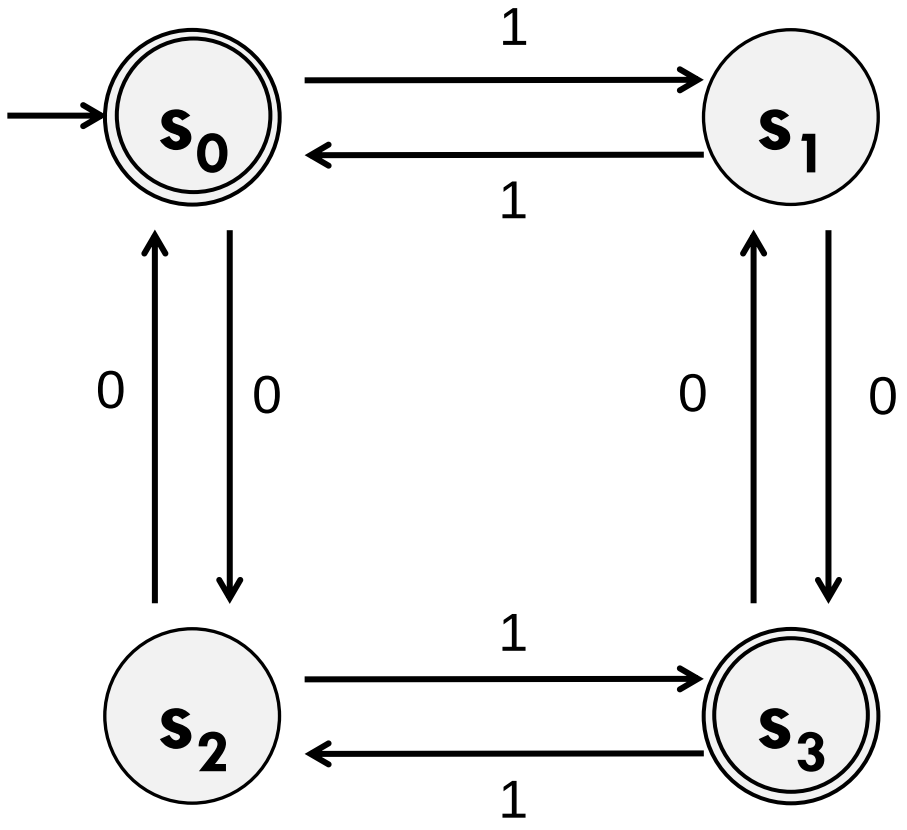
DFA Minimization

What language does this DFA recognize?



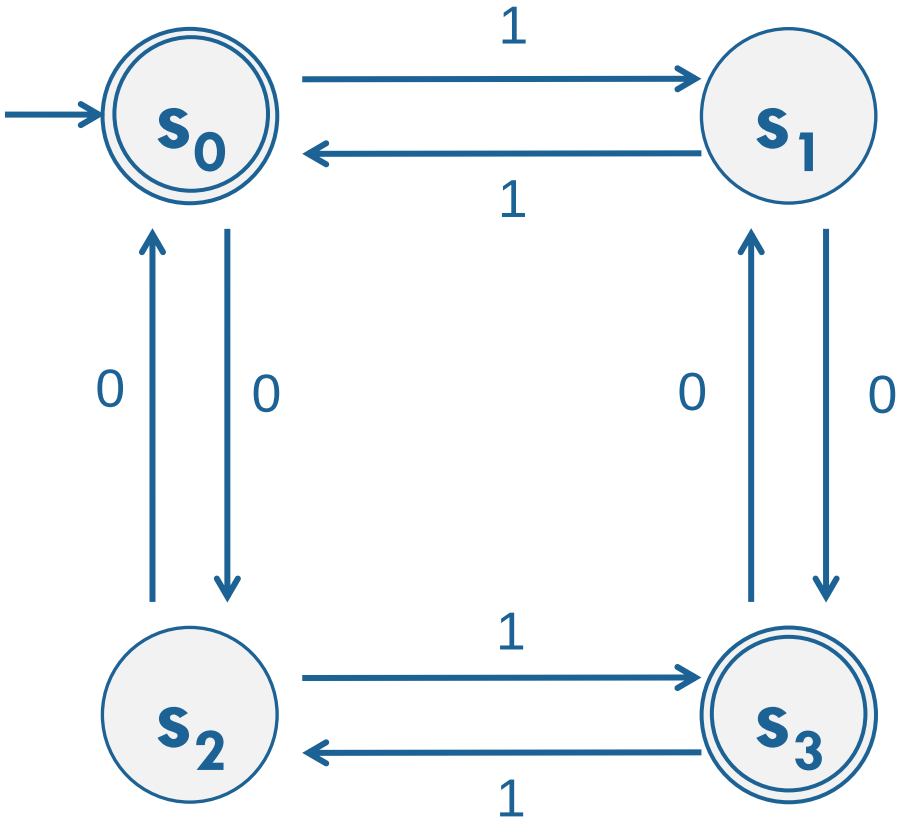
DFA Minimization

Is there an easier way to describe that?



DFA Minimization

Accepts binary strings with even # of 0s and even # of 1s, or odd # of 0s and odd # of 1s.



Accepts binary strings of even length.

These DFAs accept the same language!

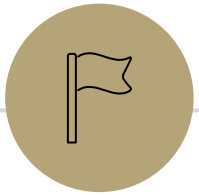
DFA Minimization

Lessons:

- The first DFA you come up with might not be the simplest.
- We aren't looking for the simplest, as long as your DFA is correct. However, simpler ideas are less prone to error.

CS Theory Results:

- There is a **unique** minimized DFA for every language.
- There is an algorithm to convert any DFA to its minimized form. (We won't cover that algorithm this quarter.)



Limitations of DFAs

Limitations of DFAs

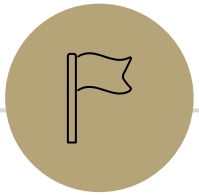
- The ultimate question for each model of computation is: Which set of languages can the model recognize?
  - Can DFAs recognize all Regular Languages?
  - Can DFAs recognize all Context Free Languages?
- We'll answer this question concretely soon. For now, here's a hint:

  There is no DFA that can recognize the language $\{0^n 1^n : n \geq 0\}$

  The intuition is that there's no way for a finite number of states to track how many 0s there have been.

Upcoming

- Friday: We'll study one more model of computation, called an NFA, which will *appear* to be more powerful than the DFA
- Monday: We'll analyze with proof exactly which set of languages (Regular, Context-Free, etc.) DFAs and NFAs can recognize.
- Wednesday: Limitations of Computation
  - We'll learn how to **prove** that a language is not Regular.
  - We'll prove that there is a language that **no computer** can recognize.



Extra Practice

Extra Practice

Consider the language “all binary strings with an even number of 0’s and each 0 is (immediately) followed by at least one 1.”

- Construct a Regular Expression that matches this language.
- Construct a DFA that recognizes this language.
- Construct a CFG that generates this language.

Extra Practice

Consider the language “all binary strings with an even number of 0’s and each 0 is (immediately) followed by at least one 1.”

Construct a Regular Expression that matches this language.

Extra Practice

Consider the language “all binary strings with an even number of 0’s and each 0 is (immediately) followed by at least one 1.”

Construct a DFA that recognizes this language.

Extra Practice

Consider the language “all binary strings with an even number of 0’s and each 0 is (immediately) followed by at least one 1.”

Construct a CFG that generates this language.