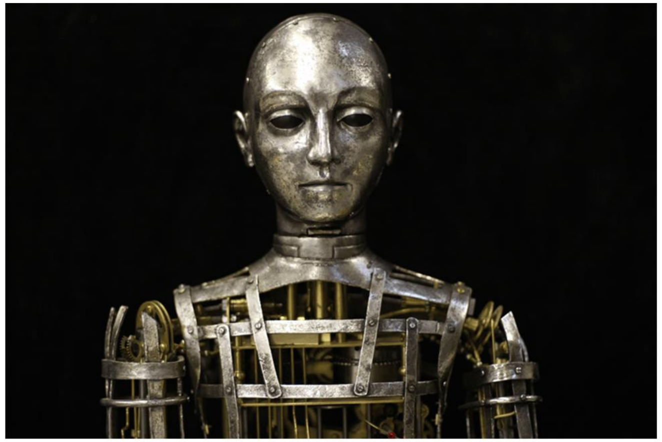
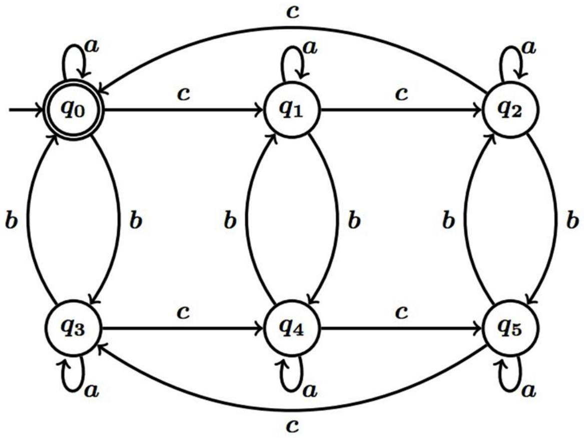
Me: Can I have automata?



CS: We have automata at home.



Nondeterministic Finite Automata (NFA)

CSE 311: Foundations of Computing I
Lecture 22

Announcements

- HW7 due Wednesday, 8/16 at 11:59 pm on Grin
  - A demo video of how to use Grin is linked in the HW writeup
  - No Late Days permitted
- Midterm solutions are at the front

Final Exam

- All information posted on Exams page of the course website.
- Final is in two parts:
  - **Part 1:** Thursday, August 17th in section Thursday (1 hour)
    Covers Lecture 1 – Lecture 16 (includes Strong Induction & Contradiction)
  - **Part 2:** Friday, August 18th in class Friday (1 hour)
    Covers Lecture 17 – Lecture 24
- Closed note, closed book. Same 3 reference sheets will be provided as the midterm.
- One practice final and solutions are posted
- Optional review session this Tuesday, August 15th from 3:00 – 4:20pm in DEM 104. Will be recorded on Panopto.

Final Exam Problem Types

Part 1:

- **General Concepts** – 1 or 2 true/false, multiple choice, or short-response problems.
- **Induction Proof** – 1 strong or weak induction proof
- **Other Proof** – 1 other proof (e.g. direct, contrapositive, cases, contradiction, biconditional)

Part 2:

- **General Concepts** – 1 or 2 true/false, multiple choice, or short-response problems.
- **Structural Induction Proof** – 1 structural induction & recursively-defined sets problem
- **Models of Computation** – 1 constructing Regular Expressions, CFGs, DFAs, and/or NFAs problem

Nondeterministic Finite Automata

NFAs

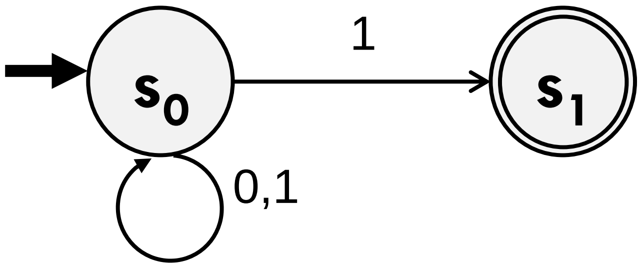
Review

- We have discussed two classes of languages: Regular Languages and Context Free Languages.
- We have discussed one model of computation: Deterministic Finite Automata
  - DFAs read in strings and accept or reject them. Each DFA recognizes a language.
- We (informally) observed that DFAs are limited
  - For example, DFAs cannot recognize the language $\{0^n 1^n : n \geq 0\}$
- Now, we will remove some of the restrictions on DFAs, to see if we can develop more powerful machine (i.e. a machine that can recognize more languages)

Nondeterministic Finite Automata

What are the components of an NFA?

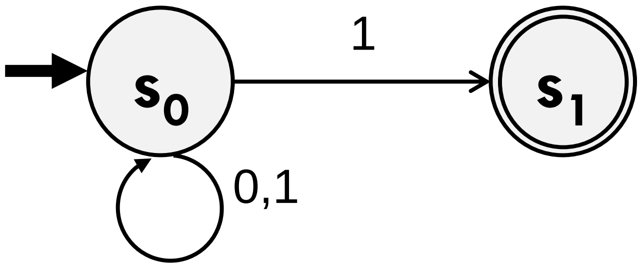
- Finite number of states
  - Accept States, Reject States, Start State (same as DFAs)
- Labelled directed edges between states, but…
  - Not required to have exactly 1 outgoing edge from each state per symbol in the alphabet. Could have 0 or >1.
  - Edges can also be labeled by the empty string ε



Nondeterministic Finite Automata

What language does an NFA recognize?

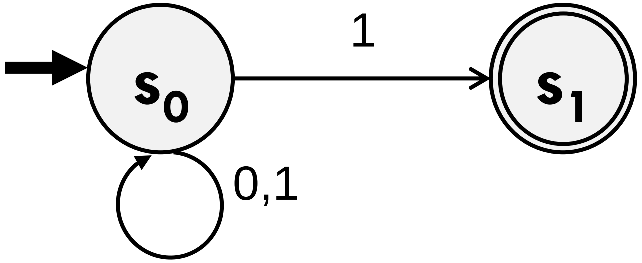
- The NFA accepts a string x iff there is **some** path from the start state to an accept state.
- For example, a single state might have several outgoing 1 edges. If you process the character "1", you can choose *any* of the outgoing 1 edges to try to reach an accept state.



Nondeterministic Finite Automata

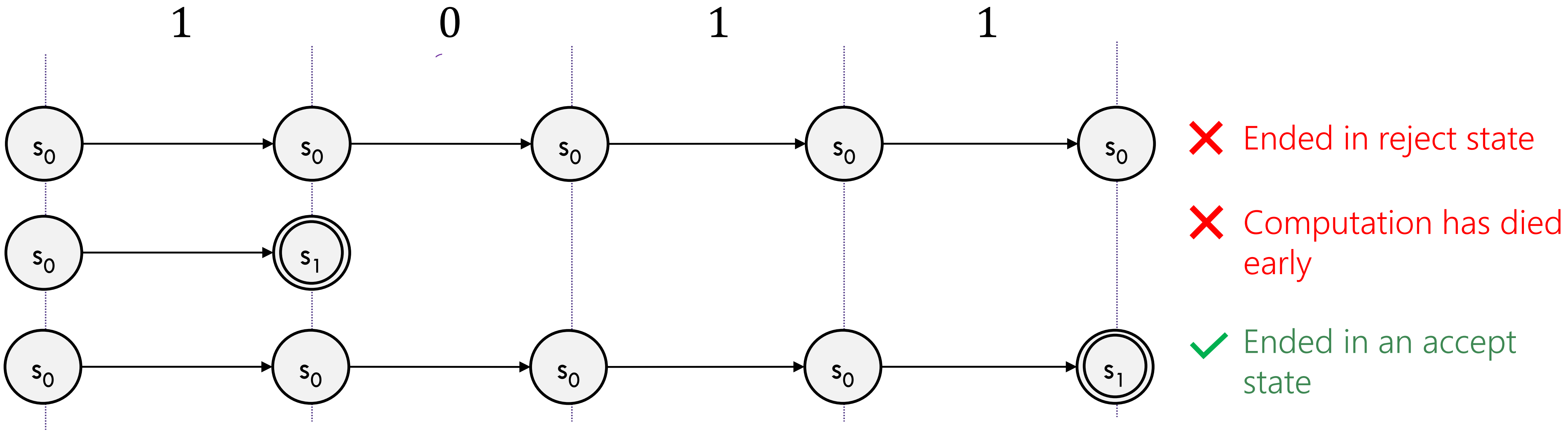
For Example:

1 0 1 1



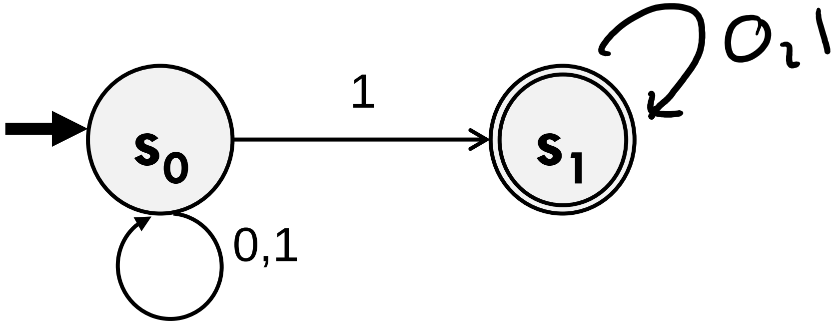
The string 1011 is accepted, because there is a path where the computation ends in an accept state!

The processing begins at s_0. From there, there are several paths through the NFA that this string could take.



Nondeterministic Finite Automata

What language does this NFA recognize?



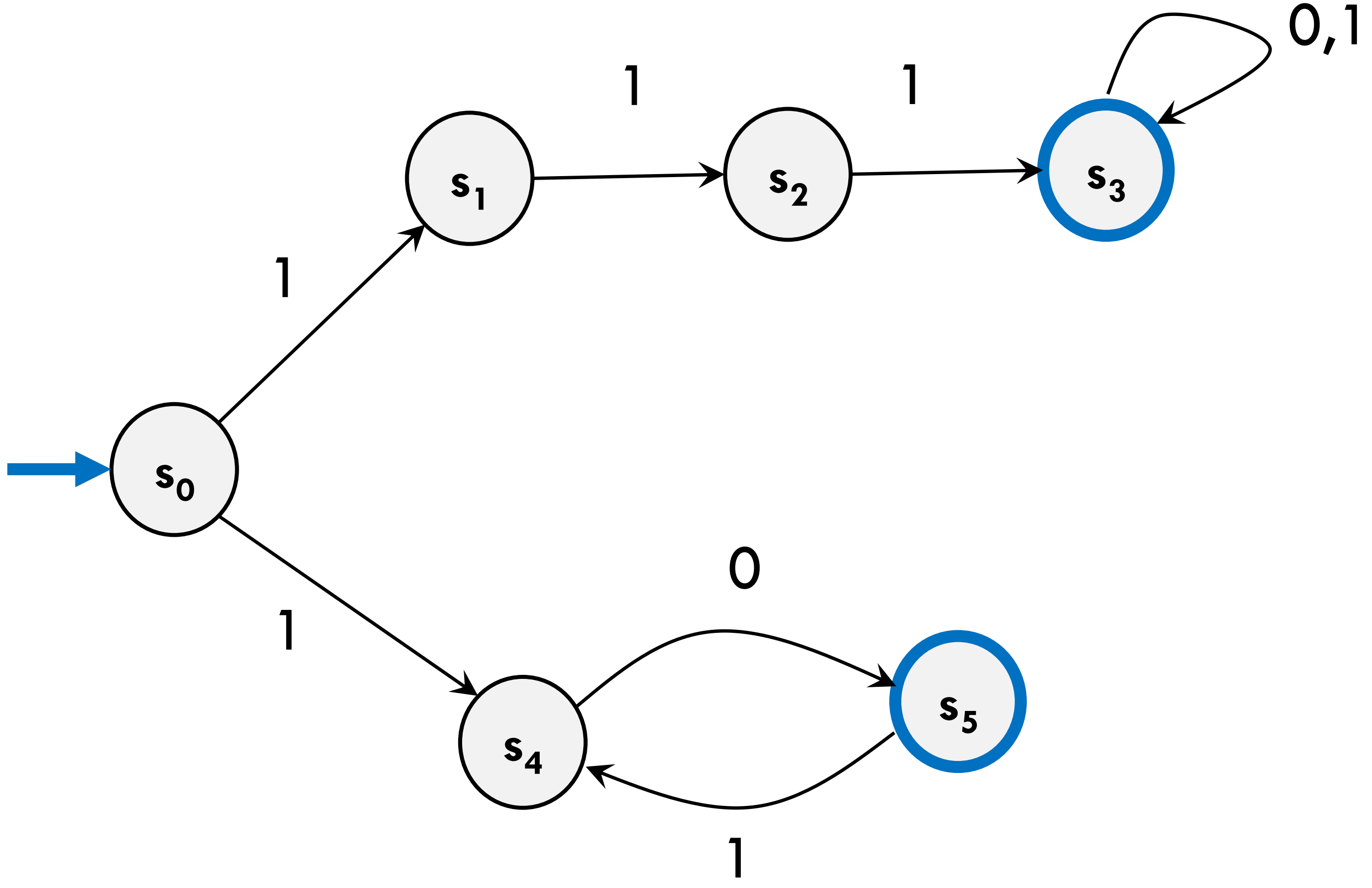
All binary strings containing a 1

All binary strings that end in a 1

Nondeterministic Finite Automata

What language does this NFA recognize?

$111(0 \cup 1)^* \cup 10(10)^*$



$111(0 \cup 1)^*$

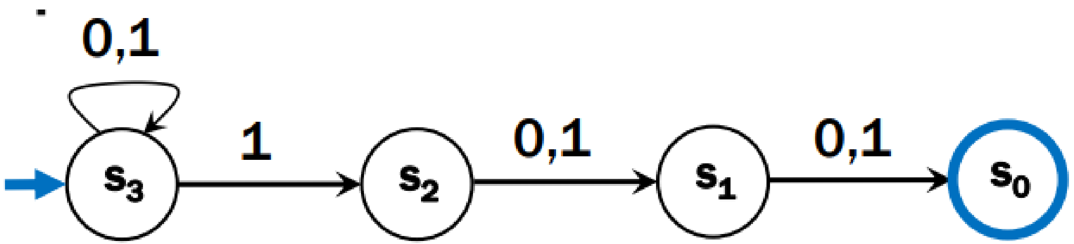
$10(10)^*$

Two ways to think about NFAs

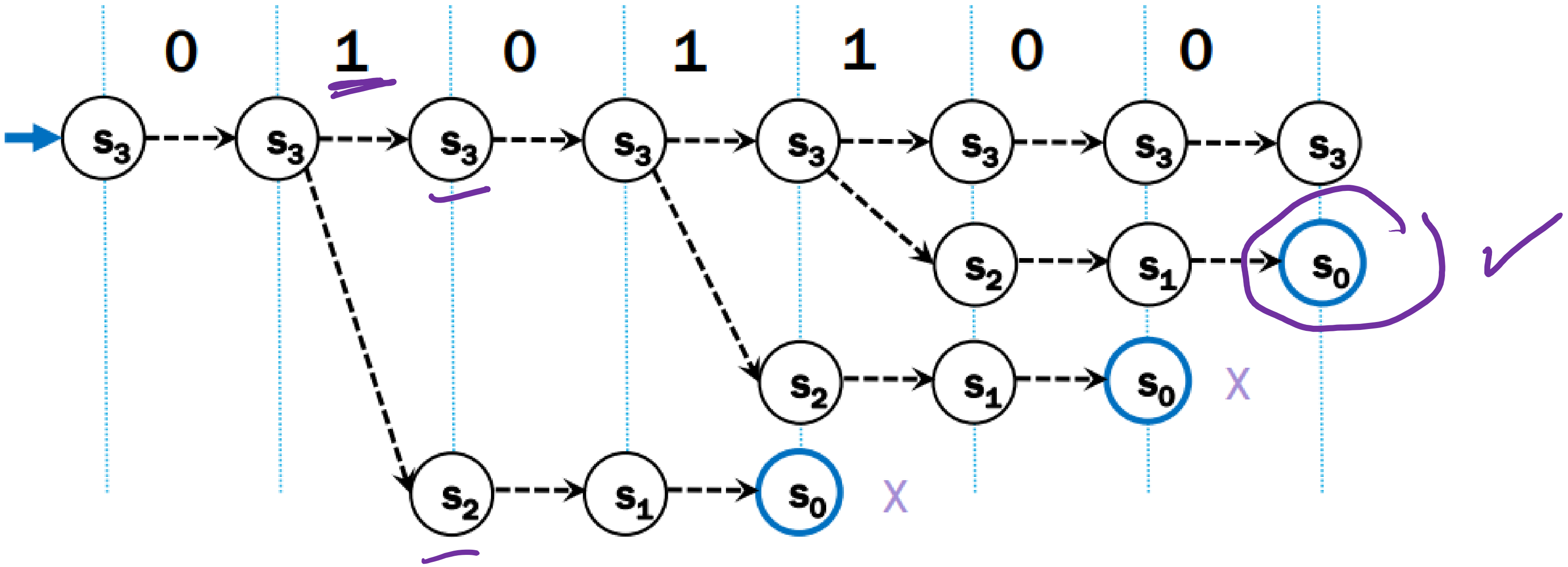
Perfect Guesser: Whenever there is a choice of what to do, the NFA correctly guesses the transition that will eventually lead to an accept state, if it exists.

Parallel Exploration: The NFA runs all possible paths that the input could take in parallel, and checks to see if any of them end in an accept state.

Parallel Exploration View of an NFA



Input string 0101100



What's with the name?

Nondeterministic Finite Automata

Nondeterministic:
Given the same input, may exhibit different behaviors on different runs.

There is more than one path a string might take through the NFA.

Finite: Not infinite; eventually ends.

An NFA has a finite number of states.

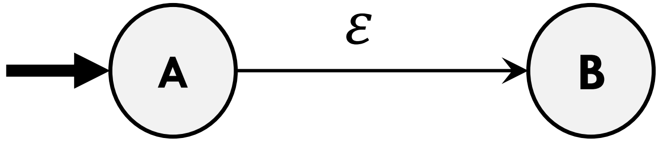
Automata: A machine that performs a function.

$P \stackrel{?}{=} NP$

NFAs with ε transitions

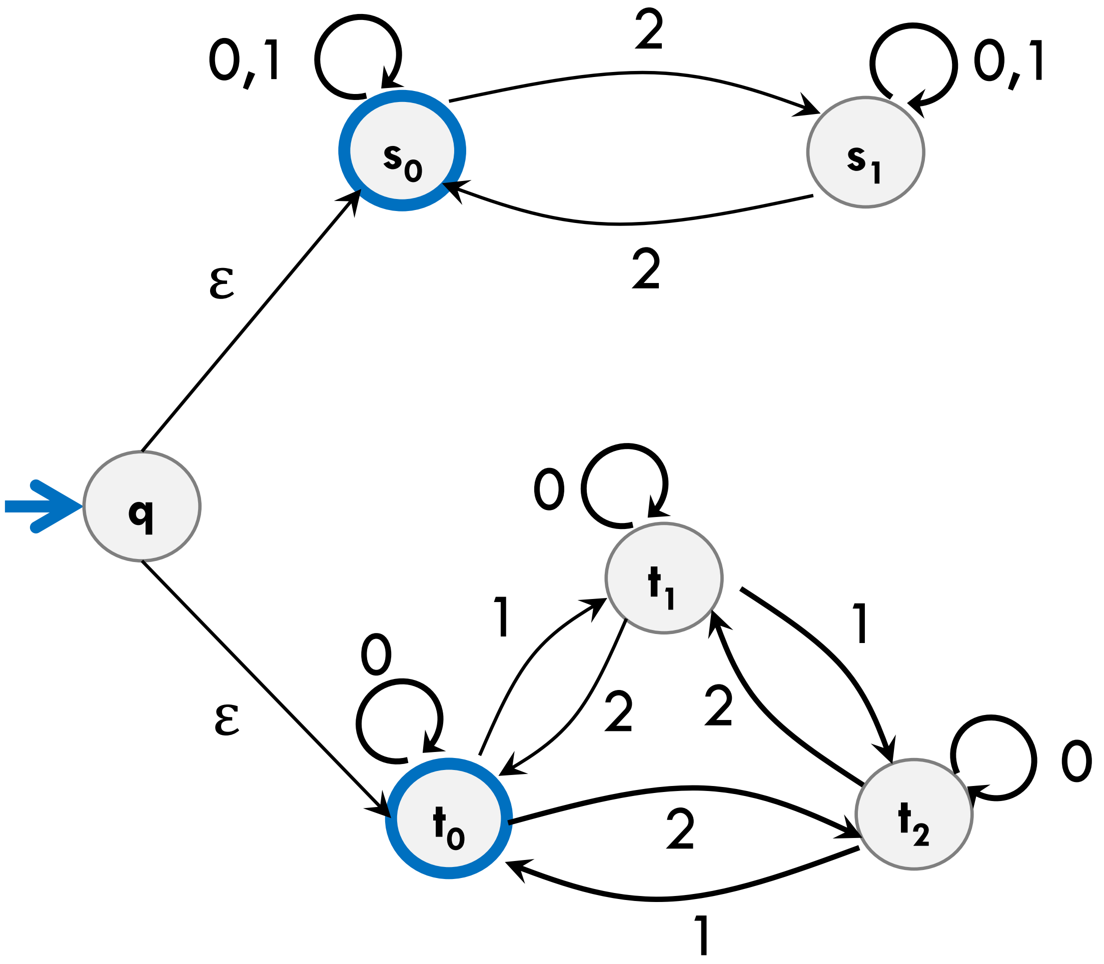
Edges labelled with ε are also known as epsilon transitions.

An ε transition from state A to state B means that the NFA can take the path from A to B without processing additional characters.



NFAs with ε transitions

What language does this NFA recognize?



Strings over {0,1,2} with an even number of 2's

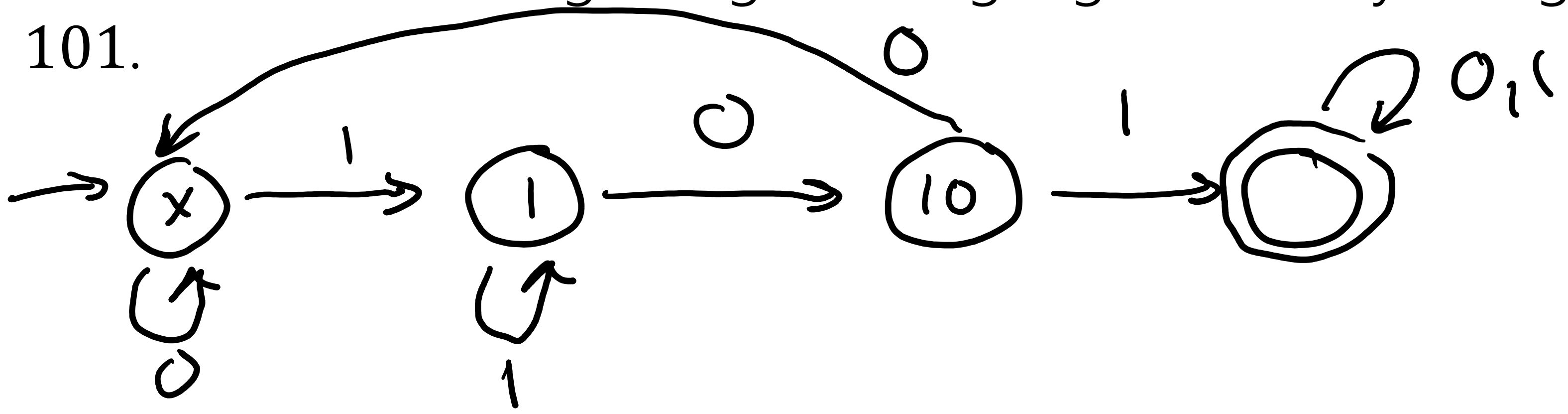
OR

Strings over {0,1,2} whose sum of digits % 3 = 0.

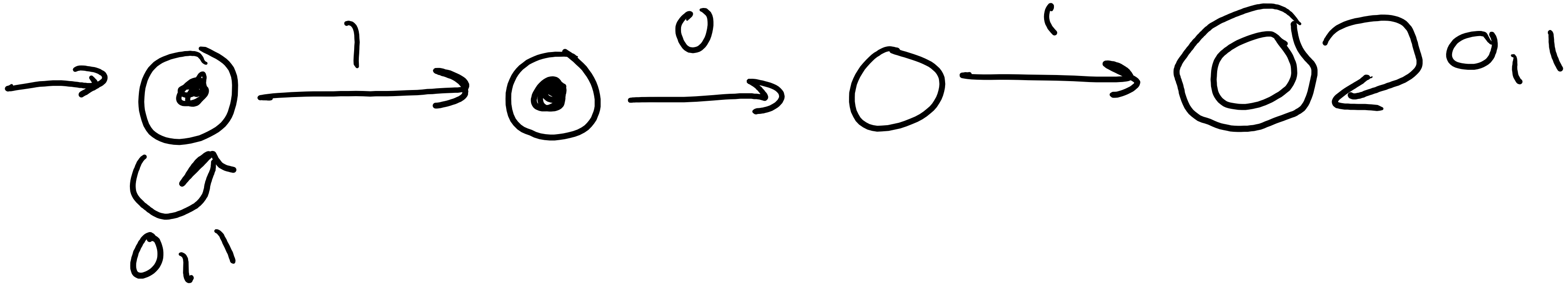
Constructing DFAs vs NFAs

Construct a DFA and an NFA recognizing the language of binary strings containing the substring 101.

DFA

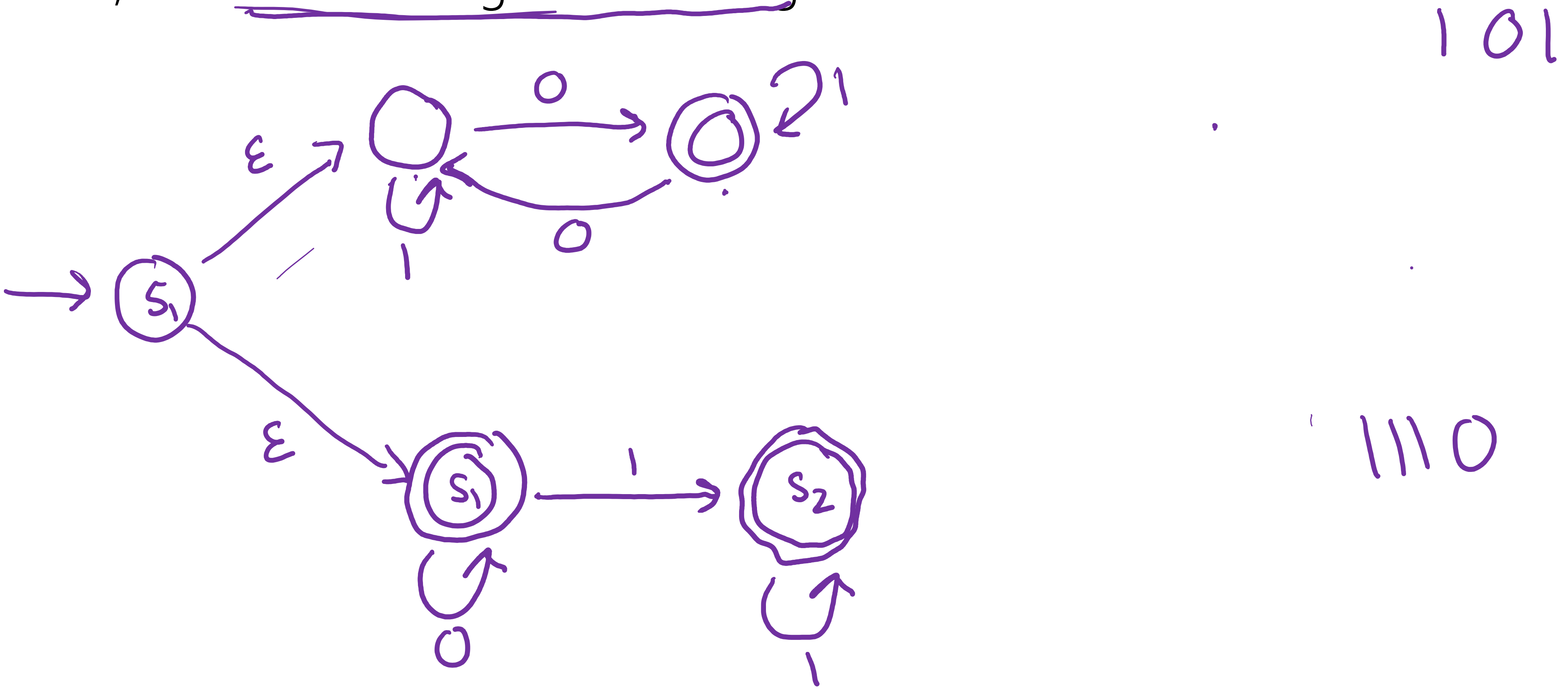


NFA



Exercise

Construct an NFA that recognizes the language of binary strings with an odd number of 0s, or not containing the substring 10.



Relationship between DFAs & NFAs

Relationship between DFAs & NFAs

DFAs have more constraints than NFAs. Every DFA is also an NFA.

It follows that every language that can be recognized by a DFA can be recognized by an NFA.

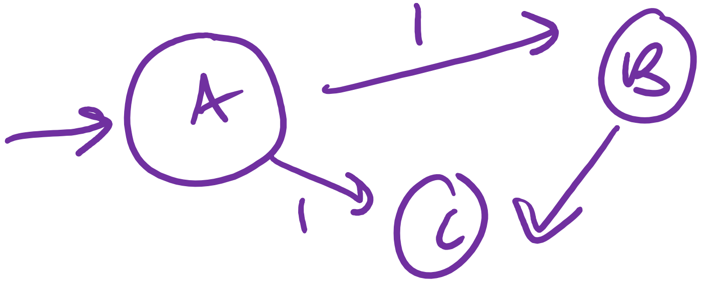
But are there languages that can be recognized by an NFA but not by any DFA? Which?

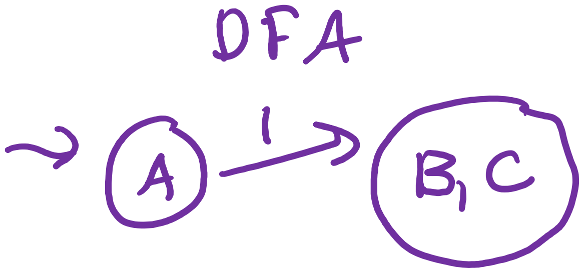
Relationship between DFAs & NFAs

Theorem: For every NFA, there is a DFA that recognizes the same language.

Corollary: The set of languages recognized by a DFA is equal to the set of languages recognized by an NFA.

Proof Idea



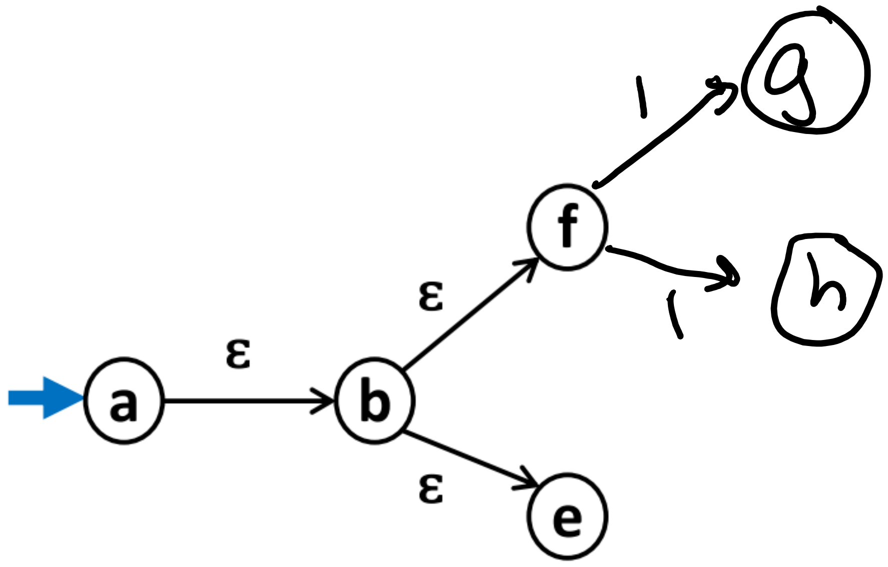


2^n

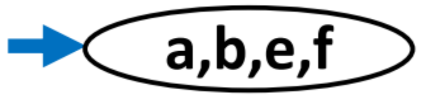
- From an NFA, we will see how to construct the corresponding DFA
- Recall the "Parallel Exploration" viewpoint of an NFA:
  The NFA runs all possible computations on the input x step-by-step at the same time in parallel.
- Construction Idea:
  - The DFA keeps track of ALL states that are reachable in the NFA along a path with some input.
  - There will be one state in the DFA for each subset of states in the NFA that can be reached by some string.

NFA to DFA Construction

Start state for the DFA: The set of all states reachable from the start state of the NFA, using only *ε* transitions.



NFA

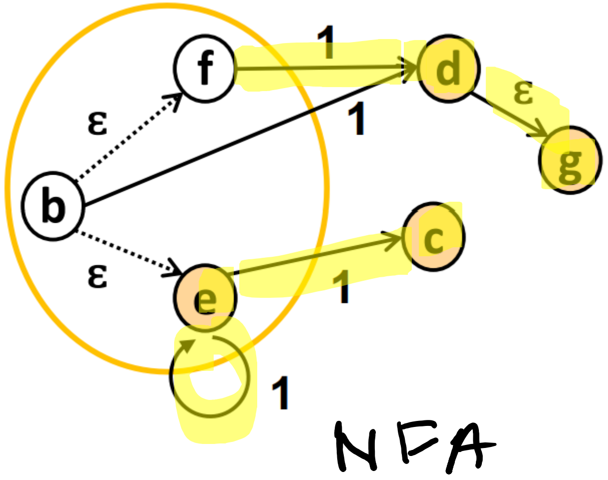


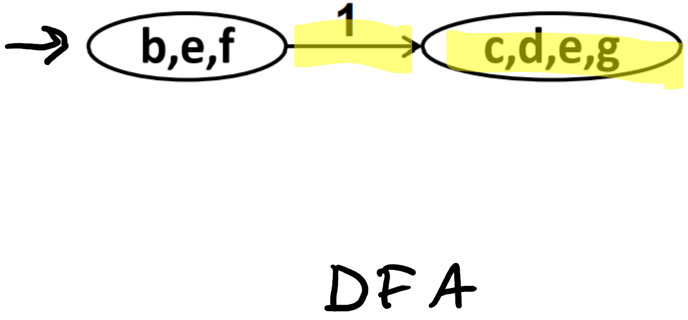
DFA

NFA to DFA Construction

Other states in the DFA:

- Choose a state in the DFA, for example the state labelled b, e, f.
- Choose a character in the alphabet, for example 1.
- Find all the states in the NFA that can be reached from b, e, f by the character 1, in this example c, d, e, g.
- Create a new state in the DFA labelled c, d, e, g, with a 1-edge from b, e, f

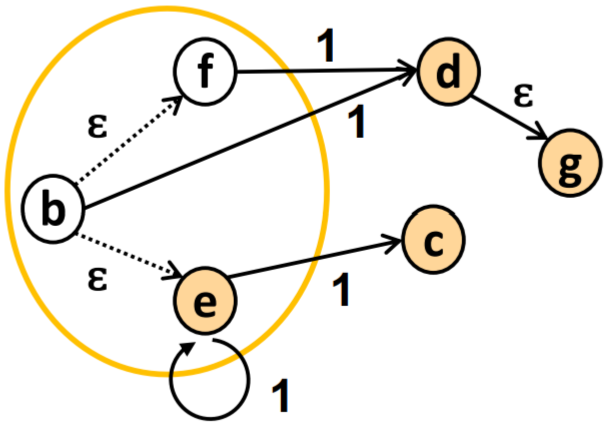


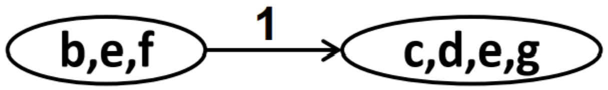


NFA to DFA Construction

Other states in the DFA: Continue this process until each state in the DFA has one outgoing edge for each character in the alphabet.

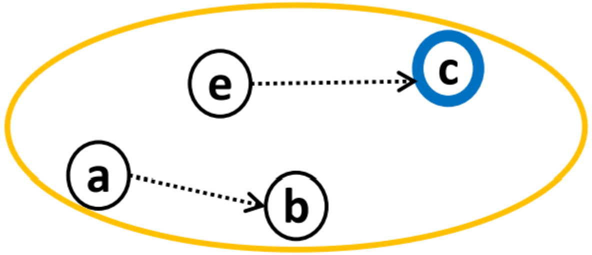
In the worst case, if the NFA had n states, the DFA will have 2^n states, since each of its states represents a subset of the original NFA's states.





NFA to DFA Construction

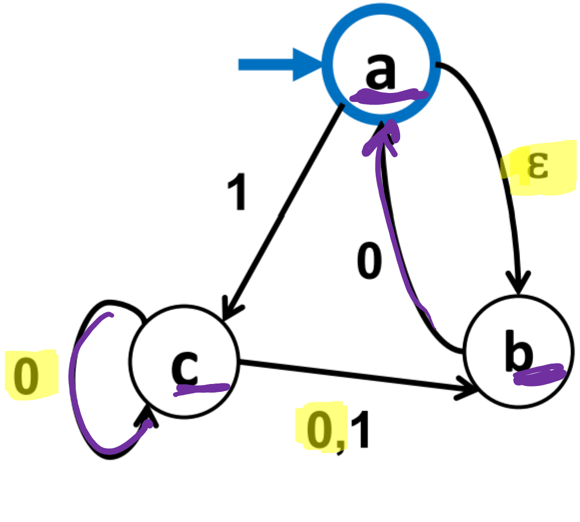
Accept states in the DFA: In the end, find all states in the DFA whose set contains an accept state in the NFA. These will be your accept states.



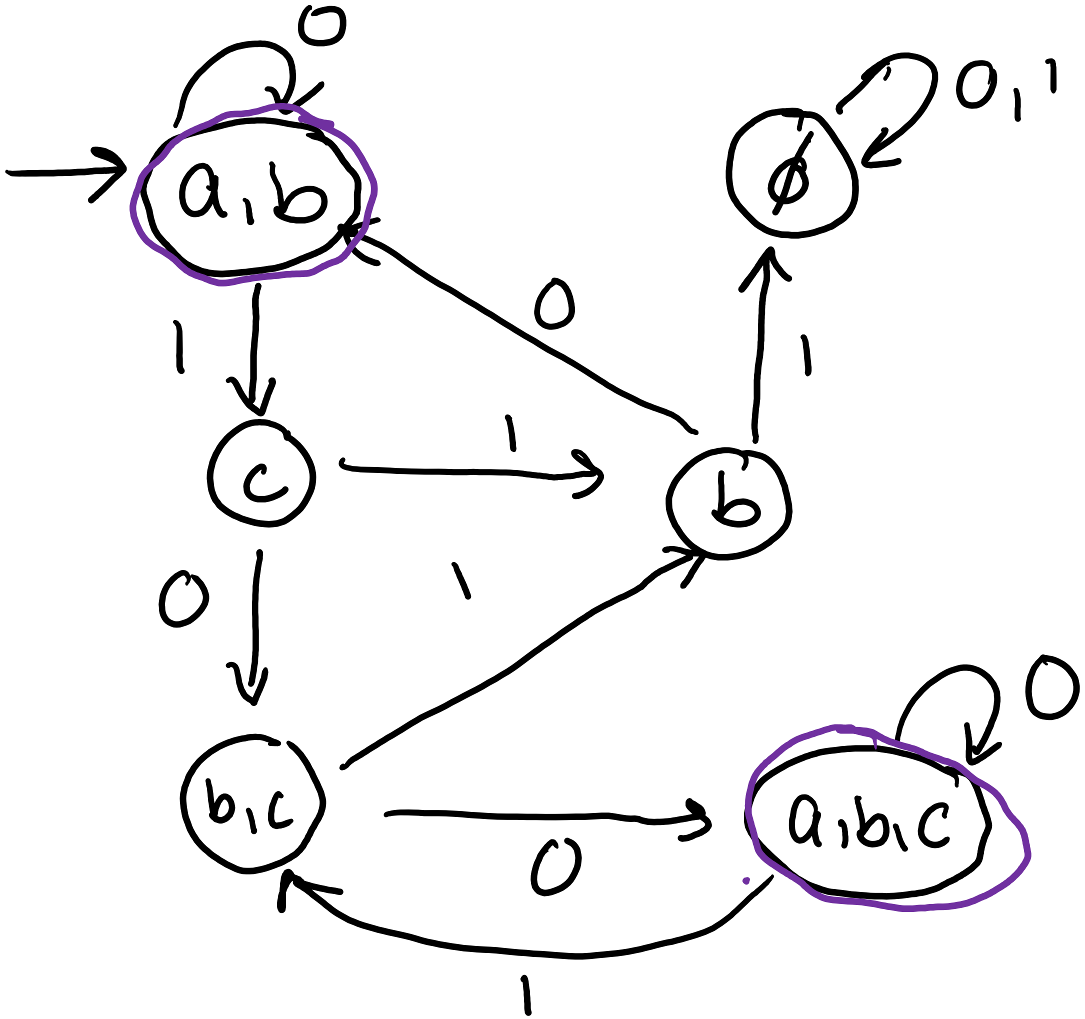
NFA

DFA

Example: NFA to DFA



NFA



Takeaways

- The set of languages recognized by DFAs is **exactly the same** as the set of languages recognized by NFAs.
- Non-determinism didn't offer a larger set of languages, but it did offer efficiency.