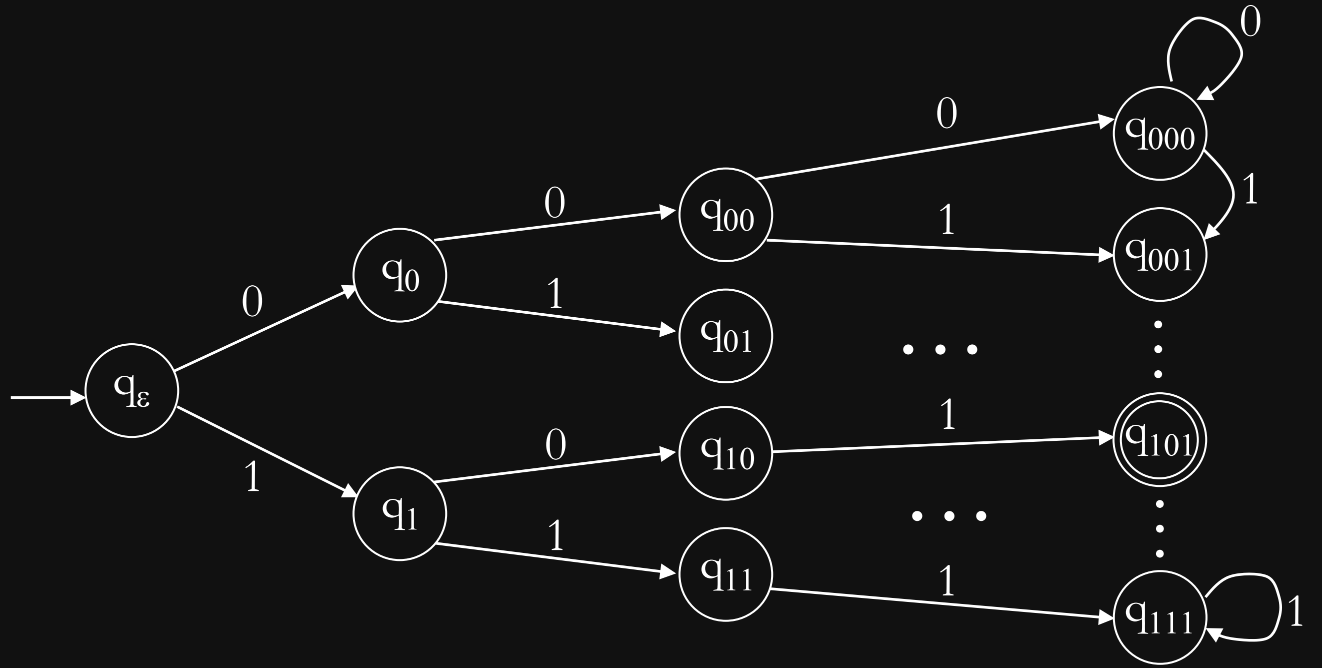
Automata theory and formal languages

Nondeterminism

- Adapted from the work of Andrej Bogdanov

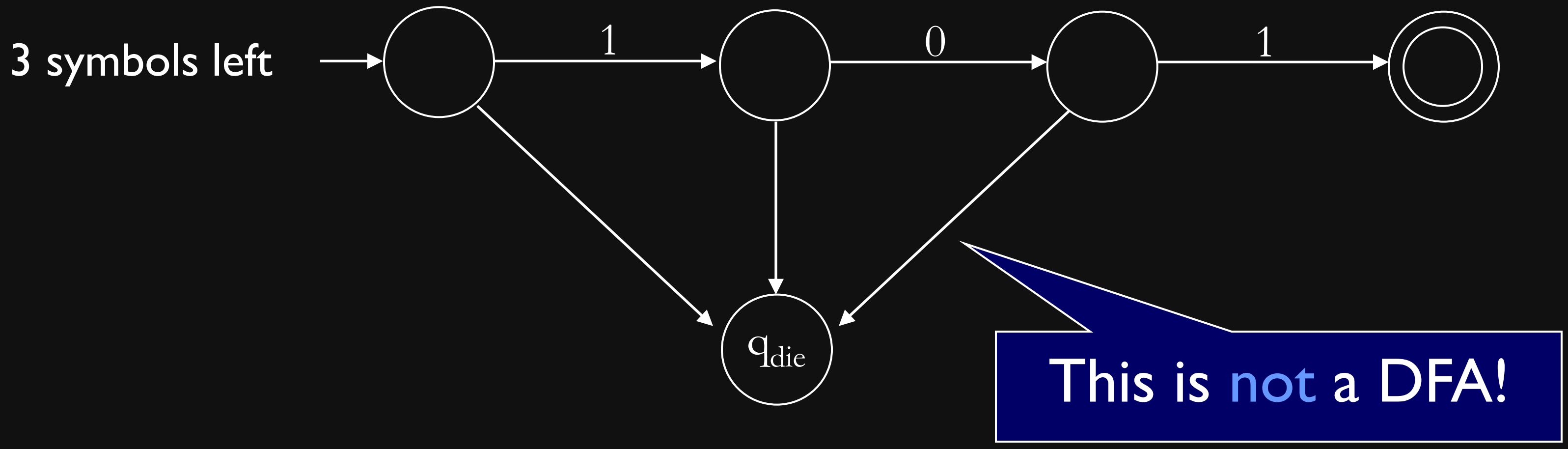
Example from last time

- Construct a DFA over alphabet $\{0, 1\}$ that accepts those strings that end in 101
- Sketch of answer:



Would be easier if…

- Suppose we could guess when the string we are reading has only 3 symbols left
- Then we could simply look for the sequence 101 and accept if we see it



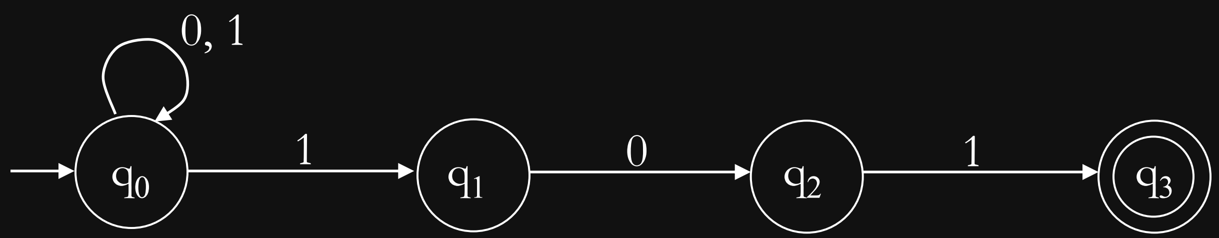
Nondeterminism

- Nondeterminism is the ability to make guesses, which we can later verify
- Informal nondeterministic algorithm for language of strings that end in 101:

1. Guess if you are approaching end of input
2. If guess is yes, look for 101 and accept if you see it
3. If guess is no, read one more symbol and go to step 1

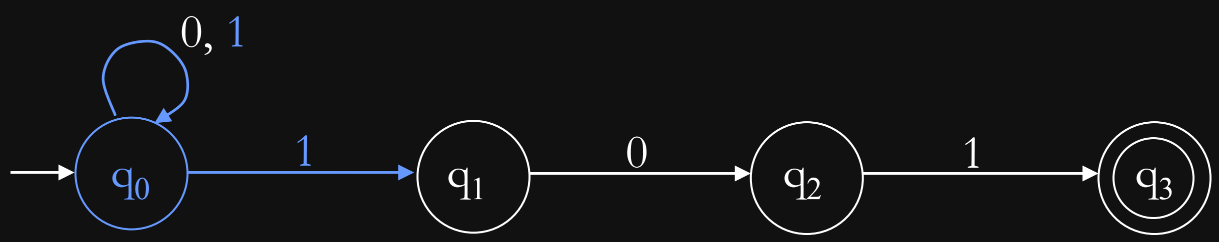
Nondeterministic finite automaton

- This is a kind of automaton that allows you to make guesses



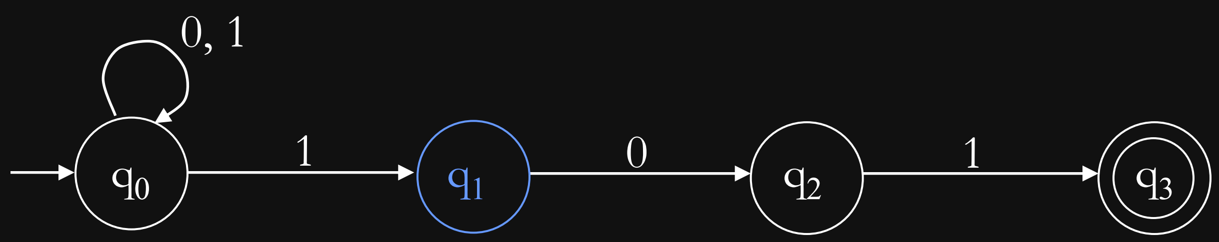
- Each state can have zero, one, or more transitions out labeled by the same symbol

Semantics of guessing



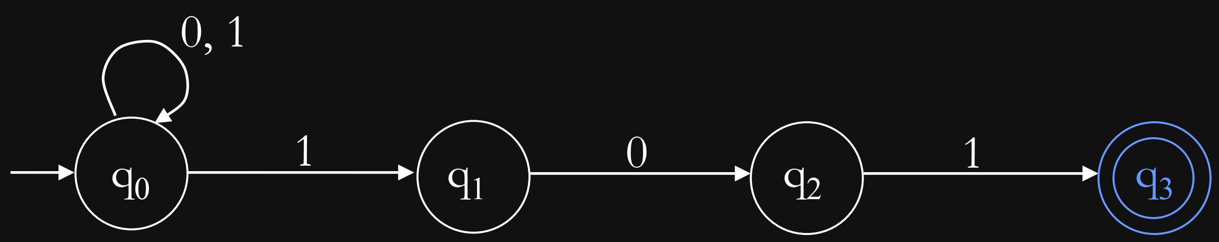
- State q_0 has two transitions labeled 1
- Upon reading 1, we have the choice of staying in q_0 or moving to q_1

Semantics of guessing



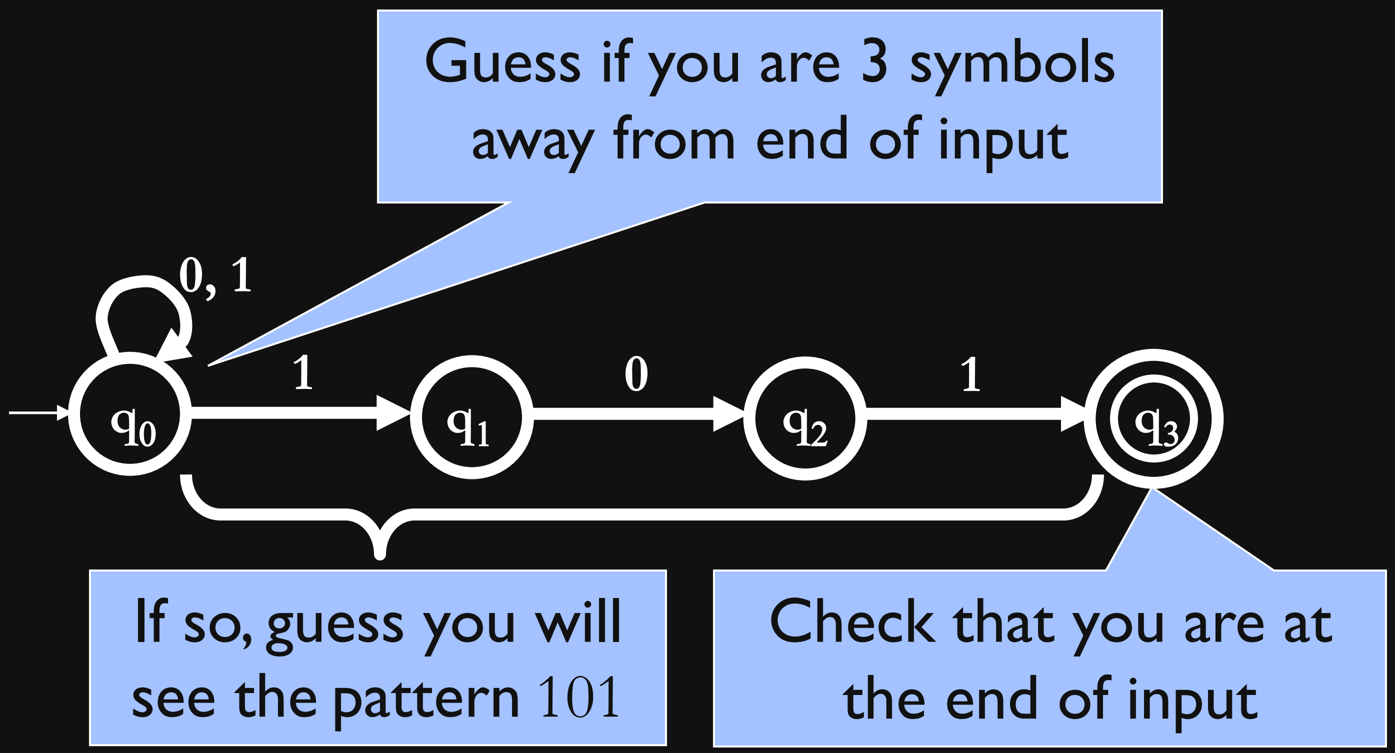
- State q_1 has no transition labeled 1
- Upon reading 1 in q_1, we die; upon reading 0, we continue to q_2

Semantics of guessing



- State q_3 has no transition going out
- Upon reading anything in q_3, we die

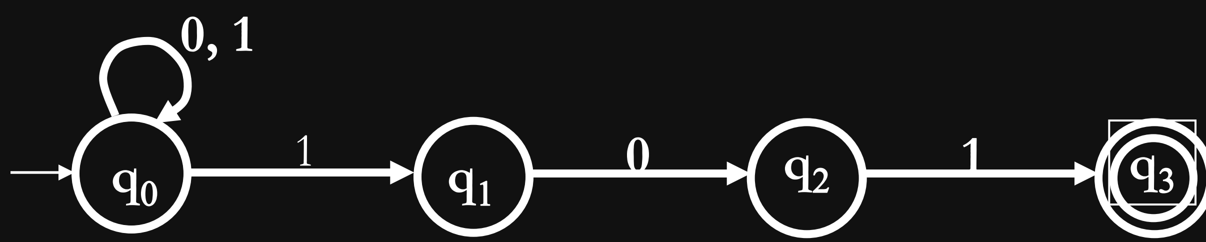
Meaning of automaton



Formal definition

- A nondeterministic finite automaton (NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where
  - Q is a finite set of states
  - Σ is an alphabet
  - $\delta: Q \times \Sigma \to$ **subsets of Q** is a transition function
  - $q_0 \in Q$ is the initial state
  - $F \subseteq Q$ is a set of accepting states (or final states).
- Only difference from DFA is that output of δ is a set of states

Example



alphabet $\Sigma = \{0, 1\}$
start state $Q = \{q_0, q_1, q_2, q_3\}$
initial state q_0
accepting states $F = \{q_3\}$

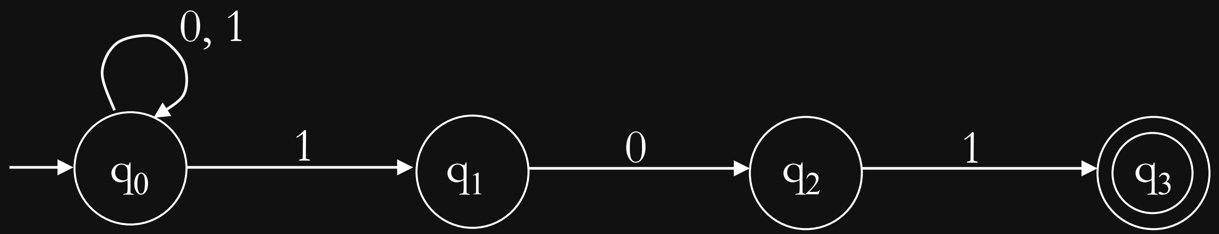
transition function δ:

	inputs	
states	0	1
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\varnothing$
q_2	$\varnothing$	$\{q_3\}$
q_3	$\varnothing$	$\varnothing$

Language of an NFA

The language of an NFA is the set of all strings for which there is some path that, starting from the initial state, leads to an accepting state as the string is read left to right.

- Example



  - 1101 is accepted, but 0110 is not

NFAs are as powerful as DFAs

- Obviously, an NFA can do everything a DFA can do
- But can it do more?

NFAs are as powerful as DFAs

- Obviously, an NFA can do everything a DFA can do
- But can it do more?

NO!

- Theorem

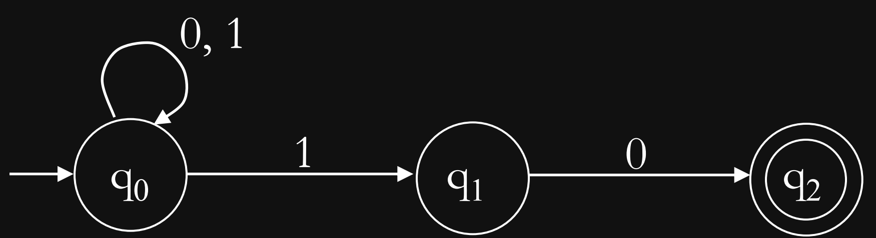
A language L is accepted by some DFA if and only if it is accepted by some NFA.

Proof of theorem

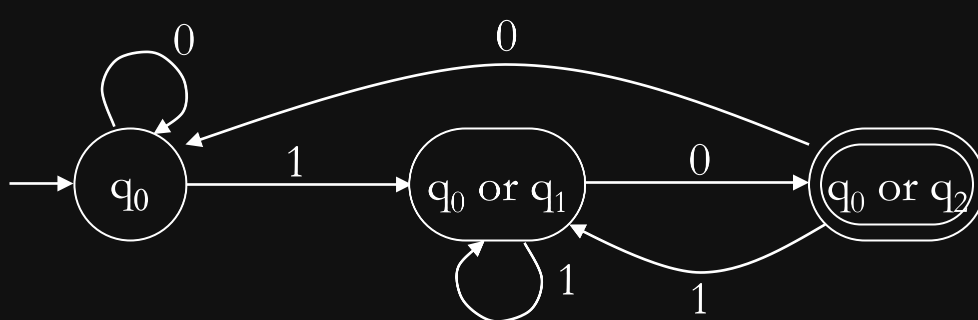
- To prove the theorem, we have to show that for every NFA there is a DFA that accepts the same language
- We will give a general method for simulating any NFA by a DFA
- Let’s do an example first

Simulation example

NFA:



DFA:



General method

	NFA	DFA
states	$q_0, q_1, \ldots, q_n$	$\varnothing, \{q_0\}, \{q_1\}, \{q_0,q_1\}, \ldots, \{q_0,\ldots,q_n\}$ one for each subset of states in the NFA
initial state	q_0	$\{q_0\}$
transitions	δ	$\delta'(\{q_{i1},\ldots,q_{ik}\}, a) =$ $\delta(q_{i1}, a) \cup \ldots \cup \delta(q_{ik}, a)$
accepting states	$F \subseteq Q$	$F' = \{S\text{: } S \text{ contains some state in } F\}$

Proof of correctness

- Lemma

After reading n symbols, the DFA is in state $\{q_{i1},\ldots,q_{ik}\}$ if and only if the NFA is in one of the states $q_{i1},\ldots,q_{ik}$

- Proof by induction on n
- At the end, the DFA accepts iff it is in a state that contains some accepting state of NFA
- By lemma, this is true iff the NFA can reach an accepting state

Exercises

- Construct NFAs for the following languages over the alphabet $\{a, b, \ldots, z\}$:
  - All strings that contain eat **or** sea **or** easy
  - All strings that contain **both** sea **and** tea
  - All strings that **do not contain** fool