Week 2

Regular Languages Pt. 2 + Grammars Pt. 1

Anakin

Outline

Nondeterminism

Context Free Grammars

Updates!

- We want feedback

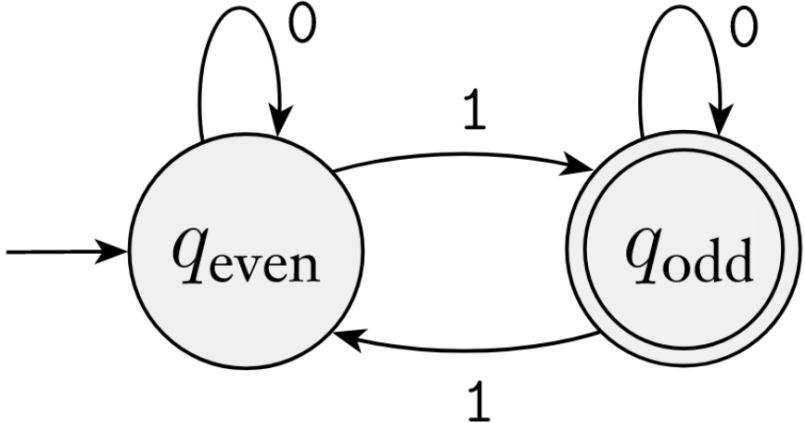
Section 1

Nondeterminism

DFA = Deterministic Finite Automata

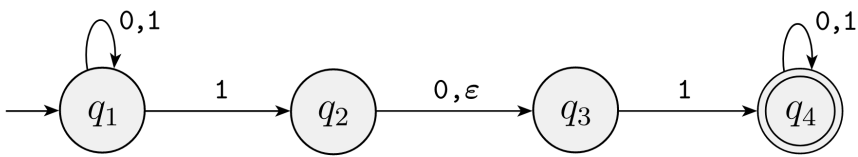
When we talked about DFAs last week, we never really delved into the name

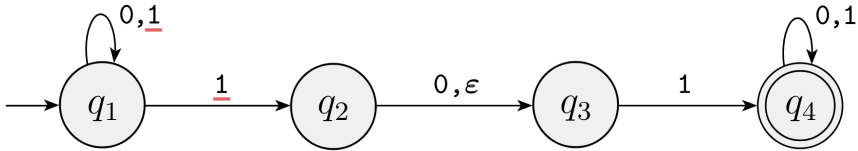
- Automata
- Finite
- Determinism?

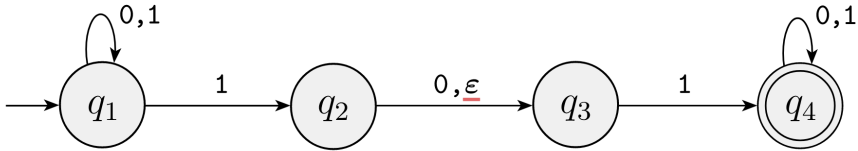

0
1
0
q_{even}
q_{odd}
1

NFA = Nondeterministic Finite Automata

What if we removed determinism?



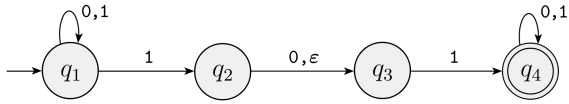

0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4


0,1
1
0,ε
1
0,1
q_1
q_2
q_3
q_4

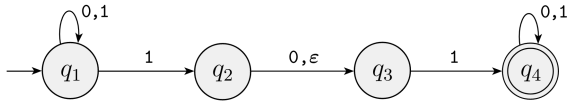
How I think of NFAs

- These are confusing at first
- Nondeterminism can be thought of in a few ways:
  - Guessing
  - Independent “processes” or “threads”
  - When computing over a string, if **any** guess is correct, the string is accepted

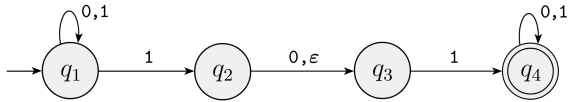
Computing on an NFA

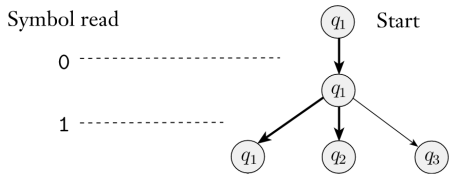


Computing on an NFA

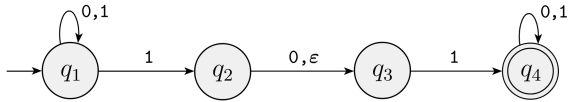


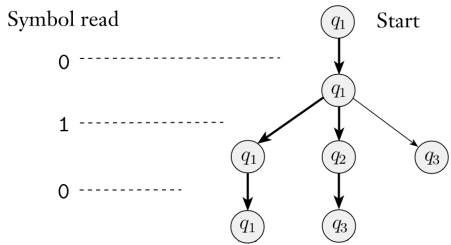
Computing on an NFA



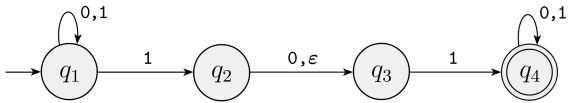


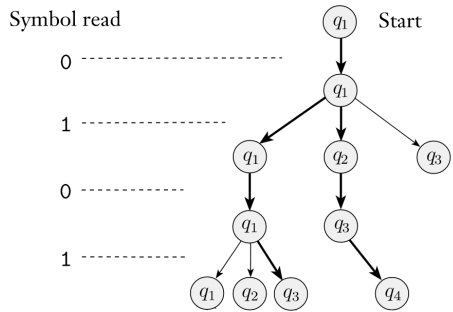
Computing on an NFA



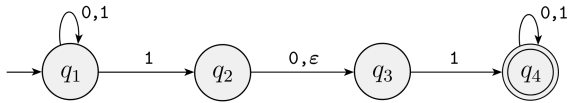


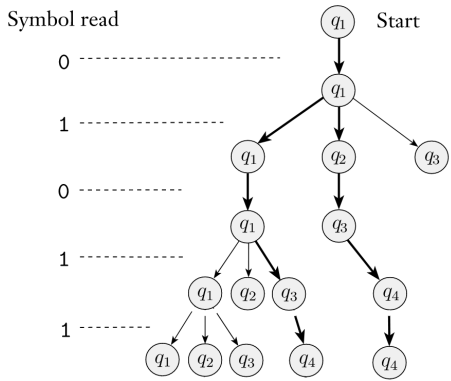
Computing on an NFA



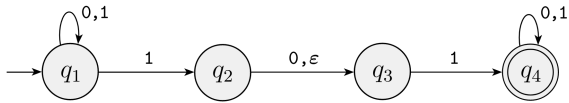


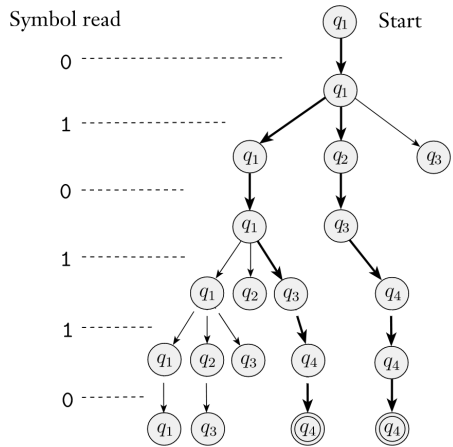
Computing on an NFA





Computing on an NFA

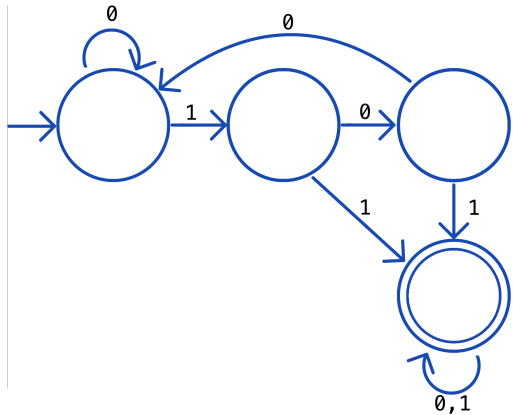




Equivalence

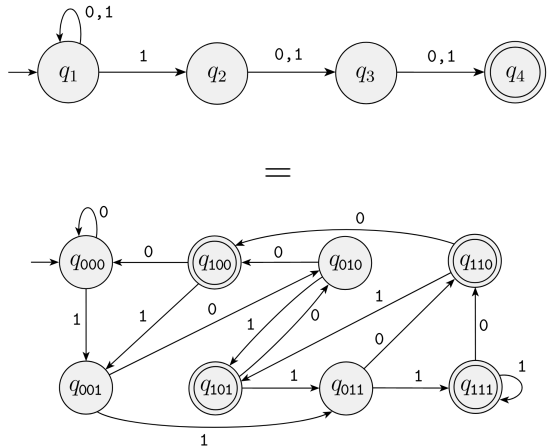
Turns out NFAs are equivalent to both DFAs and Regex

$$(0 + 1)^*(101)(0 + 1)^* + (0 + 1)^*(11)(0 + 1)^*$$



NFA & DFA Equivalence

Things aren't always that simple...



This demonstrates the power of nondeterminism!

Questions?

Questions!

Take the following description for a language and come up with a regex, NFA, and DFA for it

> "w contains an even number of `0`'s, or contains exactly two `1`'s"

Section 2

Context Free Grammars

Recursion

- DFAs / NFAs give us **iteration** and **state**
- Pure **recursion** is not emulated by DFAs / NFAs

Context Free Grammars

$$
\begin{aligned}
A &\rightarrow \mathtt{0}A\mathtt{1} \\
A &\rightarrow B \\
B &\rightarrow \varepsilon
\end{aligned}
$$

Context Free Grammars

$$
\begin{aligned}
A &\rightarrow \mathtt{0}A\mathtt{1} \\
A &\rightarrow B \\
B &\rightarrow \varepsilon
\end{aligned}
$$

Context Free Grammars

$$A \to \mathtt{0}A\mathtt{1}$$
$$A \to B$$
$$B \to \varepsilon$$

Context Free Grammars

$$
\begin{aligned}
A &\rightarrow \mathtt{0}A\mathtt{1} \\
A &\rightarrow B \\
B &\rightarrow \varepsilon
\end{aligned}
$$

Context Free Grammars

$$
\begin{aligned}
A &\rightarrow 0A1 \\
A &\rightarrow B \\
B &\rightarrow \varepsilon
\end{aligned}
$$

Context Free Grammars

$$
\begin{aligned}
A &\rightarrow \mathtt{0}A\mathtt{1} \\
A &\rightarrow B \\
B &\rightarrow \varepsilon
\end{aligned}
$$

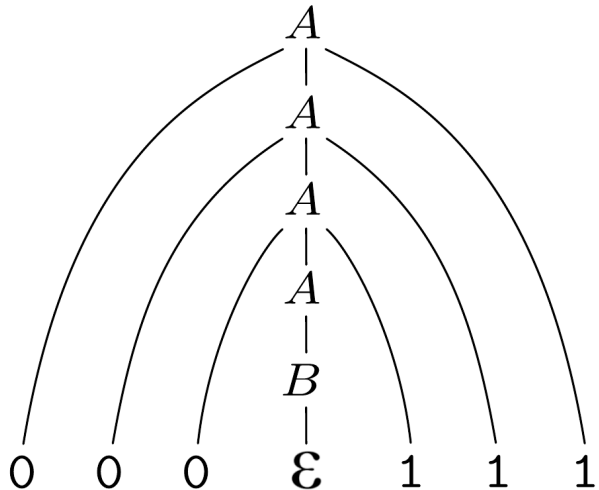
Context Free Grammars

$$
\begin{aligned}
A &\rightarrow 0A1 \mid B \\
B &\rightarrow \varepsilon
\end{aligned}
$$

Context Free Grammars

$$A \rightarrow 0A1 \mid \varepsilon$$

Is a string in a CFG?



Derivations

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\varepsilon111 \Rightarrow 000111$$

Equivalence with Regular Languages?

- CFGs define a language
- A natural question is “are these languages also regular languages?”
  - Turns out the answer is no!
  - Every regular language is context free
  - The opposite is not true. Consider $\{ \texttt{0}^n\texttt{1}^n \mid n \geq 0 \}$

Questions?

Questions!

- Come up with a CFG to match strings with twice as many `a`'s as `b`'s
- Come up with a CFG to match strings with balanced parentheses, brackets, and braces: `()`, `[]`, `{}`.

Goodbye

"You may not instantly see why I bring the subject up, but that is because my mind works so phenomenally fast, and I am at a rough estimate thirty billion times more intelligent than you. Let me give you an example. Think of a number, any number."
"Er, five," said the mattress.
"Wrong," said Marvin. "You see?"

— DOUGLAS ADAMS (1979)