CSE 311 Section 09

Models of Computation

Administrivia

Announcements & Reminders

- HW7
  - Was due yesterday, 2/28

- HW8
  - Due Wednesday 3/6 @ 11:59pm

- Final Exam
  - Monday 3/11 @ 2:30-4:20PM @ MGH 389
  - Logistics on the course website!
  - Fill Out Form for Conflict Exam
- Final Review Session
  - Saturday, 3/9 1:00-3:00pm G20

Context-Free Grammars

Context-Free Grammars

A context free grammar (CFG) is a finite set of production rules over:

- An alphabet Σ of “terminal symbols”
- A finite set V of “nonterminal symbols”
- A start symbol (one of the elements of V) usually denoted S

A production rule for a nonterminal $A \in V$ takes the form

- $A \rightarrow w1 \mid w2 \mid \ldots \mid wk$

Where each $wi \in V \cup \Sigma^*$ is a string of nonterminals and terminals.

Problem 2 – CFGs

Write a context-free grammar to match each of these languages.

a) All binary strings that start with 11.

b) All binary strings that contain at most one 1.

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

Work on this problem with the people around you.

Problem 2 – CFGs

a) All binary strings that start with 11.

Problem 2 – CFGs

a) All binary strings that start with 11.

Thinking back to regular expressions…

Problem 2 – CFGs

a) All binary strings that start with 11.

Thinking back to regular expressions…

$11\ (0 \cup 1)^*$

Problem 2 – CFGs

a) All binary strings that start with 11.

Thinking back to regular expressions…

$11\,(0 \cup 1)^*$

Now generate the CFG…

Problem 2 – CFGs

a) All binary strings that start with 11.

Thinking back to regular expressions…

$11\ (0 \cup 1)^*$

Now generate the CFG…

$$\begin{aligned} S &\rightarrow 11T \\ T &\rightarrow 1T \mid 0T \mid \varepsilon \end{aligned}$$

Problem 2 – CFGs

b) All binary strings that contain at most one 1.

Problem 2 – CFGs

b) All binary strings that contain at most one 1.

Thinking back to Regular expressions…

Problem 2 – CFGs

b) All binary strings that contain at most one 1.

Thinking back to Regular expressions…

$0^* (1 \cup \varepsilon)\, 0^*$

Problem 2 – CFGs

b) All binary strings that contain at most one 1.

Thinking back to Regular expressions…

$0^* \, (1 \cup \varepsilon) \, 0^*$

Now generate the CFG…

Problem 2 – CFGs

b) All binary strings that contain at most one 1.

Thinking back to Regular expressions…

$0^* \; (1 \cup \varepsilon) \; 0^*$

Now generate the CFG…

$$
\begin{aligned}
S &\to ABA \\
A &\to 0A \mid \varepsilon \\
B &\to 1 \mid \varepsilon
\end{aligned}
$$

Problem 2 – CFGs

b) All binary strings that contain at most one 1.

Thinking back to Regular expressions…

$0^* (1 \cup \varepsilon) 0^*$

Now generate the CFG…

$$\begin{aligned} S &\rightarrow ABA \\ A &\rightarrow 0A \mid \varepsilon \\ B &\rightarrow 1 \mid \varepsilon \end{aligned}$$

Alternative solution:

$$S \rightarrow 0S \mid S0 \mid 1 \mid 0 \mid \varepsilon$$

Problem 2 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

Problem 2 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

~~$\mathbf{S} \rightarrow 01\mathbf{S} \mid 10\mathbf{S} \mid 0\mathbf{S}1 \mid 1\mathbf{S}0 \mid \mathbf{S}01 \mid \mathbf{S}10 \mid 2$~~

Counter example: 001121100

Or:

$\mathbf{S} \rightarrow 2\mathbf{T} \mid \mathbf{T}2 \mid \mathbf{ST} \mid \mathbf{TS} \mid 0\mathbf{S}1 \mid 1\mathbf{S}0$

$\mathbf{T} \rightarrow \mathbf{TT} \mid 0\mathbf{T}1 \mid 1\mathbf{T}0 \mid \boldsymbol{\varepsilon}$

Deterministic Finite Automata

Deterministic Finite Automata

- A DFA is a finite-state machine that accepts or rejects a given string of symbols, by running through a state sequence uniquely determined by the string.

- In other words:
  - Our machine is going to get a string as input. It will read one character at a time and update "its state."
  - At every step, the machine thinks of itself as in one of the (finite number) vertices. When it reads the character, it follows the arrow labeled with that character to its next state.
  - Start at the "start state" (unlabeled, incoming arrow).
  - After you've read the last character, accept the string if and only if you're in a "final state" (double circle).

- Every machine is defined with respect to an alphabet Σ
- Every state has exactly one outgoing edge for every character in Σ
- There is exactly one start state; can have as many accept states (aka final states) as you want – including none.

Problem 3 – DFAs, Stage 1

Construct DFAs to recognize each of the following languages.
Let Σ = {0, 1, 2, 3}.

a) All binary strings.

b) All strings whose digits sum to an even number.

c) All strings whose digits sum to an odd number.

Work on this problem with the people around you.

Problem 3 – DFAs, Stage 1

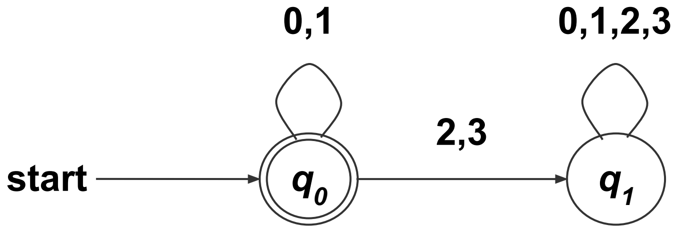
Let Σ = {0, 1, 2, 3}.

a) All binary strings.

Problem 3 – DFAs, Stage 1

Let Σ = {0, 1, 2, 3}.

a) All binary strings.



q_0 : binary strings
q_1 : strings that contain a character which is not 0 or 1

Problem 3 – DFAs, Stage 1

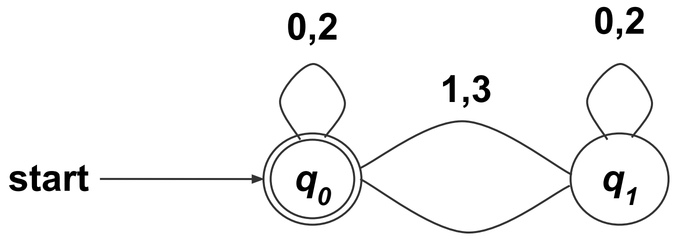
Let Σ = {0, 1, 2, 3}.

b) All strings whose digits sum to an even number.

Problem 3 – DFAs, Stage 1

Let Σ = {0, 1, 2, 3}.

b) All strings whose digits sum to an even number.



q_0: strings whose sum of digits is even
q_1: strings whose sum of digits is odd

Problem 3 – DFAs, Stage 1

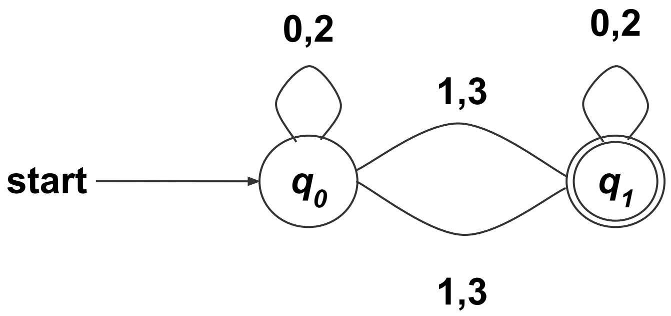
Let Σ = {0, 1, 2, 3}.

c) All strings whose digits sum to an odd number.

Problem 3 – DFAs, Stage 1

Let Σ = {0, 1, 2, 3}.

c) All strings whose digits sum to an odd number.



q_0: strings whose sum of digits is even
q_1: strings whose sum of digits is odd

Nondeterministic Finite Automata

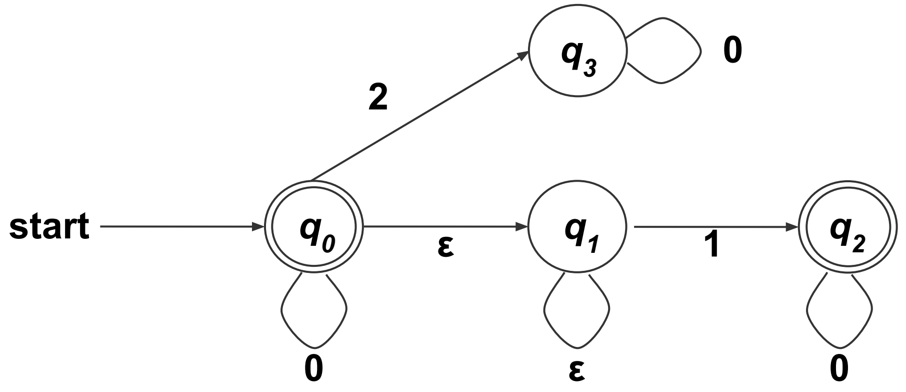
Nondeterministic Finite Automata

- Similar to DFAs, but with less restrictions.
  - From a given state, we'll allow any number of outgoing edges labeled with a given character. (In a DFA, we have only 1 outgoing edge labeled with each character).
  - The machine can follow any of them.
  - We'll have edges labeled with "ε" – the machine (optionally) can follow one of those without reading another character from the input.
  - If we "get stuck" i.e. the next character is a and there's no transition leaving our state labeled a, the computation dies.

- An NFA still has exactly one start state and any number of final states.
- The NFA accepts x if there is some path from a start state to a final state labeled with x.
- From a state, you can have 0,1, or many outgoing arrows labeled with a single character. You can choose any of them to build the required path.

Problem 5 – NFAs

a) What language does the following NFA accept?

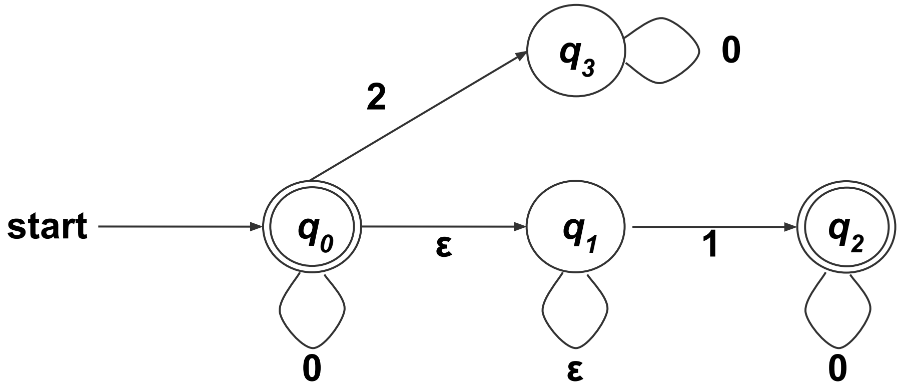


b) Create an NFA for the language "all binary strings that have a 1 as one of the last three digits".

Work on this problem with the people around you.

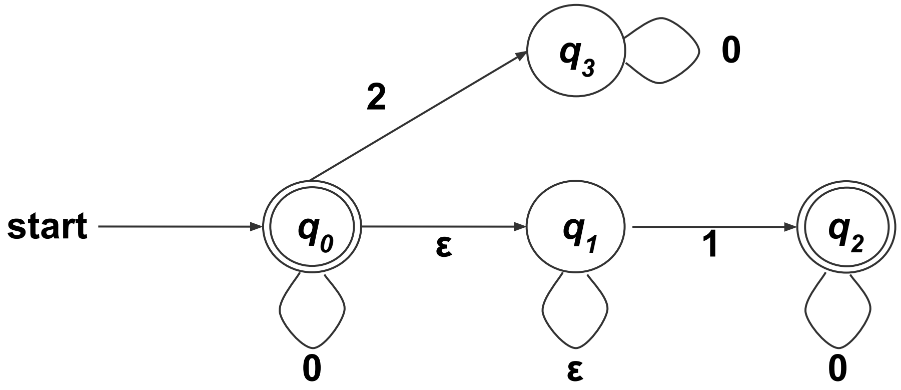
Problem 5 – NFAs

a) What language does the following NFA accept?



Problem 5 – NFAs

a) What language does the following NFA accept?



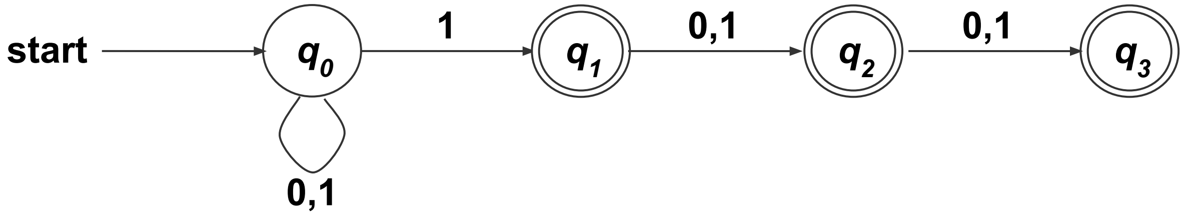
All strings of only 0’s and 1’s, not containing more than one 1.

Problem 5 – NFAs

b) Create an NFA for the language “all binary strings that have a 1 as one of the last three digits”.

Problem 5 – NFAs

b) Create an NFA for the language “all binary strings that have a 1 as one of the last three digits”.



$$\begin{tikzcd}
\text{start} \arrow[r] & q_0 \arrow[loop below, "0,1"] \arrow[r, "1"] & q_1 \arrow[r, "0,1"] & q_2 \arrow[r, "0,1"] & q_3
\end{tikzcd}$$

That’s All, Folks!

Thanks for coming to section this week!
Any questions?