CSE 311 Section 09

Models of Computation

Administrivia

Announcements & Reminders

- HW6 Regrade Requests
  - Submit a regrade request if something was graded incorrectly

- HW7
  - Due Tomorrow Friday 12/1 @10pm
  - Late due date 12/4 @ 10pm

- HW8
  - Due Friday 12/8 @ 10pm
  - Late due date 12/11 @ 10pm

- Final Exam
  - Monday 12/11 @ 4:30pm-6:20 @ KNE 130
  - Fill out Form for Conflict Exam

Recursively Defined Sets

Recursive Definition of Sets

Define a set S as follows:

Basis Step:
Describe the basic starting elements in your set
ex: $0 \in S$

Recursive Step:
Describe how to derive new elements of the set from previous elements
ex: If $x \in S$ then $x + 2 \in S$.

Exclusion Rule: Every element of S is in S from the basis step (alone) or a finite number of recursive steps starting from a basis step.

Regular Expressions

Regular Expressions

Basis:

- ε : The empty string itself matches the pattern (and nothing else does).
- $\emptyset$: No strings match this pattern
- a for any $a \in \Sigma$: The character itself matching this pattern

Recursive:

- If A, B are regular expressions then $(A \cup B)$ is a regular expression
  - matched by any string that matches A or that matches B [or both]
- If A, B are regular expressions then AB is a regular expression
  - matched by any string x such that $x = yz$, y matches A and z matches B
- If A is a regular expression, then A^* is a regular expression
  - matched by any string that can be divided into 0 or more strings that match A

Regular Expressions

A regular expression is a recursively defined set of strings that form a language.

A regular expression will generate all strings in a language, and won't generate any strings that ARE NOT in the language

Hints:

- Come up with a few examples of strings that ARE and ARE NOT in your language
- Then, after you write your regex, check to make sure that it CAN generate all of your examples that are in the language, and it CAN'T generate those that are not

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

d) Write a regular expression that matches all binary strings that do not have any consecutive 0’s or 1’s.

e) Write a regular expression that matches all binary strings of the form $1^k y$, where $k \geq 1$ and $y \in \{0,1\}^*$ has at least k 1’s.

Work on this problem with the people around you.

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

Problem 1 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0’s or 1’s.

Problem 1 – Regular Expressions

e) Write a regular expression that matches all binary strings of the form $1^k y$, where $k \geq 1$ and $y \in \{0,1\}^*$ has at least k 1's.

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

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

Problem 2 – CFGs

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

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 $\Sigma = \{0, 1, 2, 3\}$.

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 $\Sigma = \{0, 1, 2, 3\}$.

a) All binary strings.

Problem 3 – DFAs, Stage 1

Let $\Sigma = \{0, 1, 2, 3\}$.

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

Problem 3 – DFAs, Stage 1

Let $\Sigma = \{0, 1, 2, 3\}$.

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

Nondeterministic Finite Automata

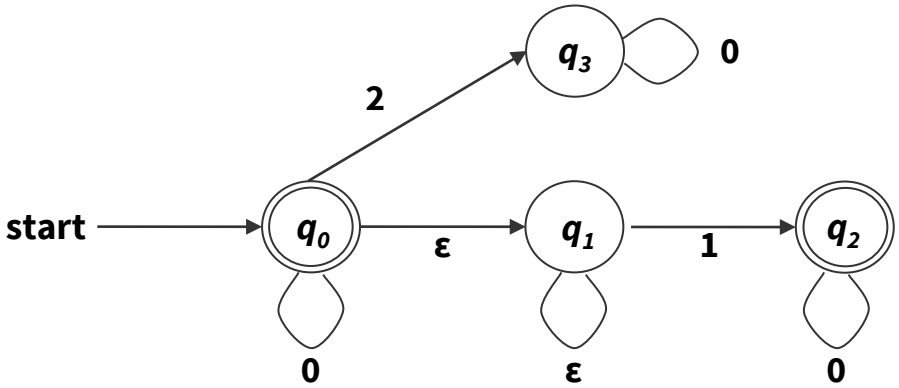
Nondeterministic Finite Automata

- 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?

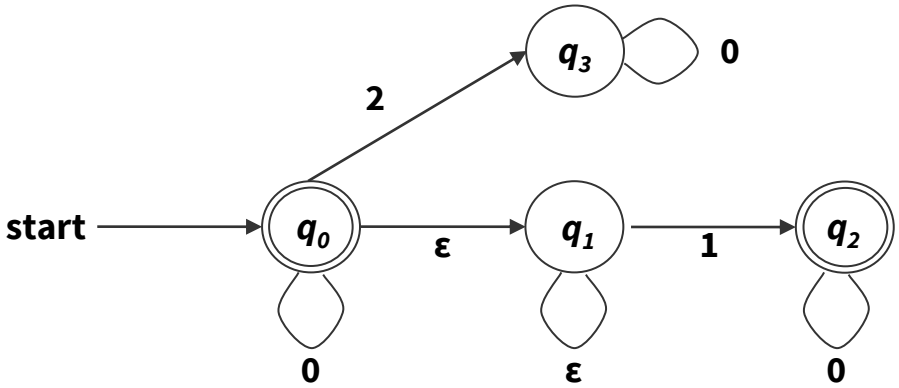


b) Create an NFA for the language “all binary strings that have a 1 as one of the last three digits”.

Work on this problem with the people around you.

Problem 5 – NFAs

a) What language does the following NFA accept?



Problem 5 – NFAs

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

That's All, Folks!

Thanks for coming to section this week!
Any questions?