COMS W3261
Computer Science Theory
Lecture 2
Finite Automata

Alexander Roth

2014-09-08

Outline

- Our languages odyssey
- Deterministic finite automata
- Examples of DFAs
- Nondeterministic finite automata
- Equivalence of DFAs and NFAs: the Subset Construction
- Bad Case for the Subset Construction

1 Our Languages Odyssey

- In this course we will study the following classes of languages in the Chomsky hierarchy:
  - The regular languages
  - The context-free languages
  - The recursive languages
  - The recursively enumerable languages
- We will discover that this is a proper hierarchy of languages.

Regular Languages Can be described using Finite Automata and Regular Expressions

Context-Free Grammars Can be used to construct Regular Languages.

Algorithm A Turing machine that halts on all inputs.

Recursively Enumerable Does not halt on all inputs.

All Languages What do you think? What kind of infinite is there?

If we look at any programming language, there are only a countably infinite number of programs; however, there are uncountably infinite languages. Thus, we reach a boundary of computing.

2 Deterministic Finite Automata

- A deterministic finite automation (DFA) is one of the simplest models of computation.
- A DFA defines a regular language.
- A DFA has five components:
  1. A finite set of states Q.
  2. An input alphabet consisting of a finite set of symbols Σ.
  3. A transition function δ that maps $Q \times \Sigma$ to Q.
     This transition function can be represented by a transition diagram in which the nodes are labeled by states and arcs by symbols.
  4. A start state which is one of the states in Q.
  5. A set of final (or accepting) states F that is a subset (possibly empty) of Q.
- A DFA accepts an input string x if and only if there is a sequence of transitions from the initial state to a final state that spells out x.
- $L(D)$, the language defined by a DFA D, is the set of strings accepted by D.
- A language accepted by a DFA is called a *regular language*.

We can imagine a Finite Automata as a circuit, which contains an input tape, with a read head and an input/output system. The machine starts in one state `A`, and moves along the tape. It begins in an initial condition, and then it makes a sequence of 0 or more moves until it reaches the final state, where it then says it *accepts* the final state.

$$D = (Q, \Sigma, \delta, q_0, F)$$

- D is the DFA.
- Q is the finite set of states.

- Σ is the input alphabet.
- δ is the transition function.
- q_0 is the start state.
- F is the set of final states $F \subseteq Q$.

$$\hat{\delta} : Q \times \Sigma^* \to Q$$

where δ is $Q \times \Sigma \to Q$, that is the transition function that maps the alphabet to the finite state machine.

"For every state and for every input symbol, there is a state it maps too." Everything need to be completely specified.

3 Example 1: DFA for All Strings of 0's and 1's With an Even Number of 0's

- Consider the following DFA D with:
  1. The finite set of states $Q = \{\texttt{A, B}\}$
  2. The input alphabet $\Sigma = \{\texttt{0, 1}\}$
  3. The transition function δ as represented by the following transition table:

State	Input Symbol	
	0	1
A	B	A
B	A	B

  4. Start state A
  5. The set of final states $F = \{\texttt{A}\}$
- Given the input string 10101, D makes the following sequence of transitions started off in its start state {A}:

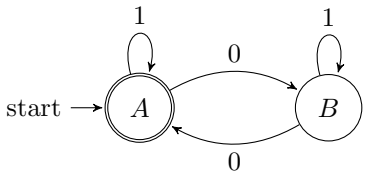
```
  1   0   1   0   1
A   A   B   B   A   A
```

  The sequence of transitions take D from its start state A to the final state B. Thus, the input 0010 is accepted by D
- Given the input string 101, D makes the following transitions started off in its initial start state A:

```
  1   0   1
A   A   B   B
```

  This sequence of transitions takes D from its start state A to the nonfinal state B. Therefore, the input 101 is not accepted by D.

- The language defined by D is the set of all strings of 0's and 1's having an even number of 0's.



$L = \{\, w |\, w$ is a string of 0's and 1's with an even number of 0's $\}$

4 Example 2: DFA for Binary Numbers Divisible by 3

- We want a DFA to recognize all strings of 0's and 1's representing binary numbers divisible by three. We assume the empty string represents 0, 0 represents 0, 1 represents 1, 00 represents 0, 01 represents 1, 10 represents 2, 11 represents 3, and so on.
- Consider the following DFA D with:
  1. The finite set of states $Q = \{\texttt{A, B, C}\}$
  2. The input alphabet $\Sigma = \{\texttt{0, 1}\}$
  3. The transition function δ as represented by the following transition table:

State	Input Symbol	
	0	1
A	A	B
B	C	A
C	B	C

  4. Start state A
  5. The set of final states $F = \{\texttt{A}\}$
- State A represents all prefixes of binary strings that are congruent to 0 mod 3, state B all prefixes congruent to 1 mode 3, and state C all prefixes congruent to 2 mod 3.
- Given the input string 1001, D makes the following transitions started off in its start state A:

```
  1   0   0   1
A   B   C   B   A
```

  In this sequence of transitions D goes from its start state A. Thus, the binary input 1001, which represents 9, is accepted by D.

$$L = \{\Sigma, 0, 11, 110, \ldots\}$$

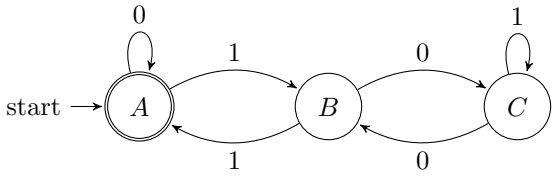
4.1 Modular Arithmetic

$$a \equiv b (\mathrm{mod}\, n) \,\mathrm{iff}\, a - b = kn$$

Thus, we have

$$5 \,\%\, 3 = 2$$

If $a_1 \equiv b_1 (\mathrm{mod}\, n)$ and $a_2 = b_2 (\mathrm{mod}\, n)$ then $a_1 + a_2 \equiv (b_1 + b_2) \,\mathrm{mod}\, n$
$a_1 a_2 \equiv (b_1 b_2)\, (mod)\, n$
Prove this.

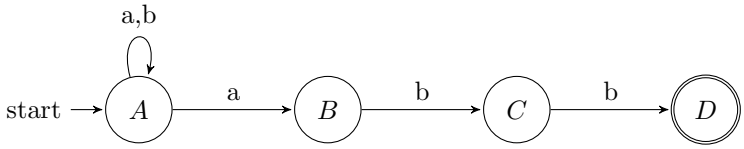


Is the finite state diagram for this machine were $A = 0 \,\mathrm{mod}\, n$, $B = 1 \,\mathrm{mod}\, n$ and $C = 2 \,\mathrm{mod}\, n$.

5 Nondeterministic Finite Automata

- A nondeterministic finite automaton (NFA) consists of
  - A finite set of states Q.
  - An input alphabet consisting of a finite set of symbols Σ.
  - A transition function δ that maps $Q \times \Sigma$ to $P(Q)$, the set of subsets of Q.
    Like a DFA, this transition function can be represented by a transition diagram in which the nodes are labeled by states and arcs by symbols. Unlike a DFA, an NFA may have transitions to zero or more states from a given state on a given input symbol.
  - A start state that is one of the states in Q.
  - A set of final (or accepting) states F that is a subset of Q.
- An NFA accepts an input string x if there is a path of transitions from the initial state to a final state that spells out x. Note that, unlike for a DFA, in an NFA, there may be more than one path from the initial state to a final state that spells out x.
- Note the definition of an extended transition function for an NFA. See Sect 2.3.3 of HMU.
- $L(A)$, the language defined by an NFA N is the set of string accepted by N.

NP vs P? Nondeterministic Polynomial time vs Polynomial time.
$L = \{\, w \mid w$ is a string of `a`'s and `b`'s ending in `abb` $\}$



A nondeterministic machine is omniscient.

6 Equivalence of DFAs and NFAs: the Subset Construction

- From an NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$, we can construct a DFA $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ that simulates all possible moves of N on any given input as follows:
  - Q_D is the set of all subsets of Q_N.
  - The input alphabet of D is the same as that of N.
  - For each state S in Q_N and each input symbol `a` in Σ. $\delta_D(S, \mathtt{a})$ is the union of all states in $\delta_N(\mathtt{p}, \mathtt{a})$ for all `p` in S.
  - The start state of D is the state $\{\ \mathtt{q}_0\ \}$.
  - F_D, the final states of D, are those states of Q_D that contains a state in F_N.
- We can prove by induction on the length of an input string `w` that D can reach deterministic state S in Q_D on `w` iff N can reach each nondeterministic state `p` in S on `w`. See Sect. 2.3.5 of HMU.

For the example of last section:

State	a	b
{A}	{A, B}	{A}
{A, B}	{A, B}	{A, C}
{A, C}	{A, B}	{A, D}
{A, D}	{A, B}	{A}

7 Bad Case for the Subset Construction

- Let L be the language $(a+b) * a(a+b)^{n-1}$, that is, all strings of a's and b's in which the n^{th} character from the end is an "a". The smallest DFA that accepts L must have at least 2^n states.

We have an NFA N that we want to transform into a DFA D such that $L(N) = L(D)$. Thus, the states on the DFA are going to be subsets of the states used in the NFA.

8 Practice Problems

1. Let L be the language { `w` | `w` is any string of `a`'s and `b`'s containing at least one `a` and at least one `b` }.
   (a) Construct a DFA for L.
   (b) Show the behavior of your DFA processing the input string `aabaa`.
2. Let L be the language { `abxba` | `x` is any string of `a`'s, `b`'s, and `c`'s not containing `ba` }. This language models comments in the programming language C.
   (a) Construct a DFA for L.
   (b) Show the behavior of your DFA processing the input string `abcbaba`.
3. Let L be the language consisting of all strings of `a`'s and `b`'s containing the substring `abba`. Construct a DFA for L.
4. Let L be the language consisting of all strings of `0`'s and `1`'s having an even number of `0` and an even number of `1`'s.
   (a) Construct a DFA for L.
   (b) Show the behavior of your DFA processing the input string `01100101`.
   (c) Prove that your DFA is correct.
5. Construct an NFA that accepts all strings of `a`'s and `b`'s ending in `abb`.
   (a) Show all sequences of moves that your NFA can make on the input string `ababb`
   (b) Use the subset construction to convert your NFA into an equivalent DFA.
6. Construct an NFA that accepts all strings of `a`'s and `b`'s in which the third last character is an `a`.
   (a) Use the subset construction to convert your NFA into an equivalent DFA.
   (b) Prove that any DFA recognizing this language must have at least eight states.

9 Reading Assignment

- HMU: Ch. 2