CSC301

Lecture 3 summary

1910456

What is a Language?

Java, English, C++ is a language.

Different languages have different symbols.

A SET OF SYMBOLS IS AN ALPHABET. Which we use as a starting block of a language.

Computer language is $\Sigma = \{0, 1\}$ (Binary Language)

String is when we take symbols from set Σ and arrange it sequentially. So, STRINGS ARE SEQUENCES. "azqq", "khbcca" are STRINGS OVER $\Sigma = \{a, b, c, \ldots, z\}$

How does this become a language?

To become a language, its needs to be over Σ but we MUST HAVE A CRITERIA.

Strings over $\Sigma = \{0, 1\}$, and length = 2 are $L_1 = \{00, 01, 10, 11\}$

Therefore,

LANGUAGE IS A SET OF STRINGS OVER AN ALPHABET AND A CRITERIA.

$L_{\text{all}} = \Sigma^* = \{\varepsilon, 0, 1, 00, 01, \ldots\}$ is the set of all possible strings over the same alphabet.

Deterministic Finite Automata

We saw a graphical representation of how a gumball machine operates.

States are represented by circles. \$0, \$5, \$10.

Start State has an arrow going into it (from nowhere).

Final State(s) have another circle within the circle. It is an "accepting" state.

The automata takes inputs of {+5, +10, R}.

Transition arrows MUST emerge from each state FOR EVERY ALPHABET SYMBOL. They tell us what happens.

Formally Defining Finite Automata

A DFA is a 5-TUPLE $(Q, \Sigma, \delta, q_0, F)$

- Must have, Q, a finite SET of states.

- Must operate over Σ, an alphabet (SET of symbols).
- What to do, after a state over Q takes an alphabet (action) over Σ as input is defined in δ, the transition FUNCTION where $\delta : Q \times \Sigma \to Q$
- Mandatory initial state is q_0 where $q_0 \in Q$.
- F is the SET of accepting states (final states) where $F \subseteq Q$.

There can be no final states as well.

After all actions (inputs) are taken and we end up in a final state. Then we can say the sequences of actions are the "right" sequences of actions.

Transition Functions can be drawn, written, or be represented as a table.

To know if an automata diagram is a DFA, we see if the automata has all the requirements written above.

Example images in https://imgur.com/a/NNSmNXf

Example 1

There are 3 states, Q = {q_0, q_1, q_2}

Alphabets are Σ = {0, 1}

The initial state is q_0.

The final states are F = {q_0, q_1}

For the transitions, δ is:

(q_0, 0) → q_0

(q_0, 1) → q_1

(q_1, 0) → q_2

(q_1, 1) → q_1

(q_2, 0) → q_2

(q_2, 1) → q_2

But since δ is cumbersome to list like this, we use a table. (Shown in link)

	0	1
→ q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_2	q_2

Does example 1 accept "0001"?

We will consider each of the symbols as an action.

We will take inputs in this fashion state «0 0 0 1.

Therefore, we start from q_0 which consumes 0 and so on.

$(q_0, 0001) \rightarrow (q_0, 001) \rightarrow (q_0, 01) \rightarrow (q_0, 1) \rightarrow (q_1, \varepsilon)$

After all the actions are taken, we are at state q_1 which is an accepting state.

$q_1 \in F$, therefore, the DFA "accepts" this string.

Does example 1 accept "101"?

$(q_0, 101) \rightarrow (q_1, 01) \rightarrow (q_2, 1) \rightarrow (q_2, \varepsilon)$

$q_2 \notin F$, so example 1 does not accept this string.

Does example 1 accept Σ^* ? NO

Does example 1 accept $L_1 \subset \Sigma^*$? If yes, then L_1 IS THE LANGUAGE of example 1.

LANGUAGE of EXAMPLE 1 is the set of all strings that do not have a "10" as substring.

Example 2

----

	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_0

LANGUAGE of EXAMPLE 2 is the set of all strings that ends with "a".

Example 3

----

	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_1	q_2
q_2	q_1	q_2
q_3	q_4	q_3
q_4	q_4	q_3

LANGUAGE of EXAMPLE 3 is the set of all strings that has:

(Prefix "a" AND Suffix "a") OR (Prefix "b" AND Suffix "b")

Example 4

It is a DFA that accepts at most three 1s.

For ε to be part of the language, the initial state must be an accepting state.

Example 5

It is a DFA that has only {010, 1} as a language over $\Sigma = \{0,1\}$

How did we design this?

We see the strings in our language. Then, we only think about what the valid input is. Once all the actions are taken and we are left with ε, we mark the last state as an accepting state.

Any action after "010" is not in the language so we send all subsequent inputs to the q_{die} state.

The DFA also accepts "1", so we directly go to the accepting state after taking action "1". Again, subsequent inputs are sent to state q_{die}.

Lastly, when there is a breach of sequence like "011", "110" etc. we go to q_{die}.

Example 6

We design a language with Σ^* of length 2.

Up to q_2, we have taken 2 actions.

q_2 is the final state and the state we are in after any action of length <2 || >2 is not an accepting state.

Example 7

A DFA over alphabet {0,1} that accepts all strings that have a suffix "01".

Algorithmic approach shown, it has a lot of branches. There should be a better way.

Example 8

A DFA over alphabet {0,1} that accepts all strings that have a suffix "101".

Following the algorithmic approach, the number of states and branches will be too much.

We will follow the "intuitive" approach taken in Example 5.

Following that, we arrive at a much simpler diagram which is also easy to read.

Mr. Zulker noted,

In the algorithmic approach, we "remember" the suffix through the states.

In the "intuitive" approach, we "remember" the suffix through the arrows that originated from q_0.