

Regular Languages II: Nondeterministic FSAs

LING 106, February 18-25, 2009

1. REVIEW: (NON-)DETERMINISTIC FSAS

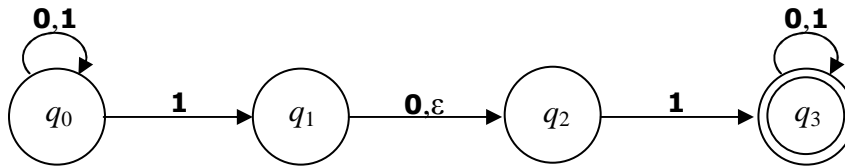
- In a deterministic FSA (DFA), all moves are uniquely determined. That is, for any state q , each symbol of the alphabet will have exactly one transition from q to another state q' (which may be the same as state q).

The transition function δ is a function $Q \times \Sigma \rightarrow Q$.

- In a non-deterministic FSA (NFA), not all moves are uniquely determined. A state may have zero, one, or more than one transitions for each symbol of the alphabet. In addition, we'll allow NFAs to transition with the empty string ϵ .

δ is a mapping, $Q \times (\Sigma \cup \{\epsilon\}) \rightarrow Q$, but it may not be a function.

1.1. A sample DFA: N_1



δ for N_1 :

- $\langle q_0, \mathbf{0} \rangle \rightarrow q_0$
- $\langle q_0, \mathbf{1} \rangle \rightarrow q_0$
- $\langle q_0, \mathbf{1} \rangle \rightarrow q_1$
- $\langle q_1, \mathbf{0} \rangle \rightarrow q_2$
- $\langle q_1, \epsilon \rangle \rightarrow q_2$
- $\langle q_2, \mathbf{1} \rangle \rightarrow q_3$
- $\langle q_3, \mathbf{0} \rangle \rightarrow q_3$
- $\langle q_3, \mathbf{1} \rangle \rightarrow q_3$

Once again, note that this is not a function from $Q \times \Sigma \rightarrow Q$, or even $Q \times (\Sigma \cup \{\epsilon\}) \rightarrow Q$. For this machine, $\delta(\langle q_0, \mathbf{1} \rangle)$ has two possible values, and $\delta(\langle q_1, \mathbf{1} \rangle)$ has none.

δ for N_1 in table form

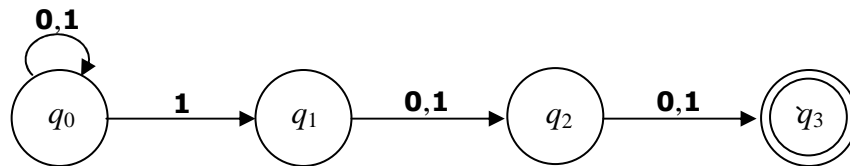
	0	1	ϵ
q_0	q_0	q_0 or q_1	UNDEFINED
q_1	q_2	UNDEFINED	q_2
q_2	UNDEFINED	q_3	UNDEFINED
q_3	q_3	q_3	UNDEFINED

Finally, at the end, there are three possible states that can be reached with the full string. We check to see if any of those are end states; if so, the FSA accepts the string, and if not, it does not. In this case, q_3 is an end state, so the string is accepted.

Question: Which of the following strings does the machine accept?

- a. **1011**
- b. **1010101**
- c. **00001**
- d. **1100101**

Question: Give a formal description of the following NFA, N_2 .



What language does N_2 define?

Question: Draw a state diagram for $N_3 = \langle Q, \Sigma, \delta, s, F \rangle$, where

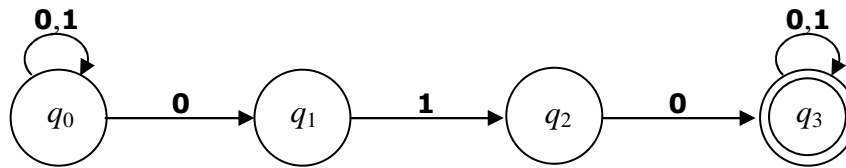
- 1. $Q = \{q_0, q_1, q_2\}$
- 2. $\Sigma = \{\mathbf{a}, \mathbf{b}\}$
- 3. $\delta =$

	a	b	ϵ
q_0	\emptyset	$\{q_1\}$	$\{q_2\}$
q_1	$\{q_1, q_2\}$	$\{q_2\}$	\emptyset
q_2	$\{q_0\}$	\emptyset	\emptyset

- 4. $s = q_0$
- 5. $F = \{q_0\}$

3. DESIGNING NFAS

- The language $\{w \mid w \text{ contains the substring } \mathbf{010}\}$, $\Sigma = \{\mathbf{0,1}\}$.



- **Question:** The language $\{w \mid \text{the length of } w \text{ is at most } 4\}$, $\Sigma = \{\mathbf{0,1}\}$.

- **Question:** The language $\{w \mid w \text{ contains an odd number of } \mathbf{1s}\}$, $\Sigma = \{\mathbf{0,1}\}$.

- **Question:** The language $\{\epsilon, \mathbf{101}, \mathbf{10001}\}$, $\Sigma = \{\mathbf{0,1}\}$.

Notation: for any string σ , the notation σ^* means “any number n of concatenated occurrences of σ , $n \geq 0$ ”. (If σ is longer than a single symbol, it’s enclosed in parentheses.)

- $\mathbf{10}^* = \mathbf{1}, \mathbf{10}, \mathbf{100}, \mathbf{1000}, \mathbf{10000}, \dots$
- $\mathbf{(10)}^* = \epsilon, \mathbf{10}, \mathbf{1010}, \mathbf{101010}, \dots$

- **Question:** The language $\{\mathbf{0}^*\}$.

- **Question:** The language $\{\mathbf{0}^*\mathbf{10}^*\}$.

- **Exercise for recitation:** Give a nondeterministic FSA diagram with the specified number of states for each of the following languages with alphabet $\{0, 1\}$:
 - a. The language $\{\epsilon\}$; one state
 - b. The language $\{0\}$; two states
 - c. The language $\{w \mid w \text{ ends with } 00\}$; three states
 - d. The language $\{001^*\}$; three states
 - e. The language $\{w \mid w \text{ starts with } 010\}$; four states
 - f. The language $\{0^*1^*0\}$; three states

4. NONDETERMINISTIC FSAS VS. DETERMINISTIC FSAS

Clearly, NFAs are more powerful than DFAs—they allow empty strings in transitions, they allow multiple transitions for the same symbol from a single state, or none at all. “More powerful”, in this case, means “can model more languages”. That is:

Proposition A: there is a language L that can be modeled with a NFA but not a DFA.

Rather than just stating the proposition with the word “clearly” in front of it, it would be nice to prove it. And yet, for all that it may look intuitively true, Proposition A is false. In fact:

Proposition B: NFAs and DFAs are equivalent: they model exactly the same languages.

The proof of Proposition B:

- (1) Show that any language modeled by a DFA can be modeled by a NFA.
- (2) Show that any language modeled by a NFA can be modeled by a DFA.
- (3) Given (1) and (2): they can model exactly the same languages.

That is: If N is the set of languages that can be modeled by an NFA, and D is the set of languages that can be modeled by a DFA, then the proof is: (1) $D \subseteq N$; (2) $N \subseteq D$; (3) $D = N$.

4.1. *If a language can be modeled by a DFA, it can be modeled by a NFA*

Proof: any DFA can be trivially converted into an NFA. Because:

- If $\mathbf{D} = \langle Q_D, \Sigma_D, \delta_D, s_D, F_D \rangle$ is a DFA, then $\mathbf{N} = \langle Q_N, \Sigma_N, \delta_N, s_N, F_N \rangle$ is an NFA that models the same language, where:
 - $Q_N = Q_D$
 - $\Sigma_N = \Sigma_D$
 - $s_N = s_D$
 - $F_N = F_D$
 - For all $q \in Q_D, x \in \Sigma_D$:
 - $\delta'(\langle q, x \rangle) = \{\delta(\langle q, x \rangle)\}$, and
 - $\delta'(\langle q, \epsilon \rangle) = \emptyset$

That was the easy part.

4.2. *If a language can be modeled by a NFA, it can be modeled by a DFA*

Once again, a proof by construction: given an arbitrary NFA, how can we construct a DFA that accepts the same strings?

- General approach: when we were computing whether a particular string was accepted by a NFA, we had to keep track, at each step, which states were reachable by the string up to that step.

Therefore: the DFA will have a state for each set of states in the NFA. That is, each state in the DFA will correspond to some subset of the states of the NFA—i.e., a member of the power set of the states.

So: let $\mathbf{N} = \langle Q_N, \Sigma_N, \delta_N, s_N, F_N \rangle$. Then $\mathbf{D} = \langle Q_D, \Sigma_D, \delta_D, s_D, F_D \rangle$ is an equivalent DFA where:

1. $Q_D = \wp(Q_N)$.
2. $\Sigma_D = \Sigma_N$, i.e. it's the same alphabet as the alphabet of N
3. For $R \in Q_D$ and $x \in \Sigma_D$, let $\delta_D(\langle R, x \rangle) = \{q \in Q_N \mid q \in \delta_N(\langle r, x \rangle) \text{ for some } r \in R\}$ ¹

...which will almost, but not quite, work: NFAs can also transition on the empty string—the above doesn't say what to do with $\delta(\langle q, \epsilon \rangle)$. So...

For any $R \in Q_D$, let $E(R)$ be the set of states that can be reached from R by going along ϵ transitions, including those states in R . That is, for $R \subseteq Q_N$,

$$E(R) = \{q \mid q \text{ can be reached from } R \text{ by following 0 or more } \epsilon \text{ transitions}\}$$

And thus: $\delta_D(\langle R, x \rangle) = \{q \in Q_N \mid q \in E(\delta_N(\langle r, x \rangle)) \text{ for some } r \in R\}$

4. $s_D = E(\{s_N\})$
that is, the start state is the one representing the start state of N and any other state that can be reached from there via empty transitions
5. $F_D = \{R \in Q_D \mid R \text{ contains an accept state of } N\}$

¹ Also written: $\delta_D(\langle R, a \rangle) = \bigcup_{r \in R} \delta(\langle r, a \rangle)$

4.3. Example

[see class slides, to be posted]

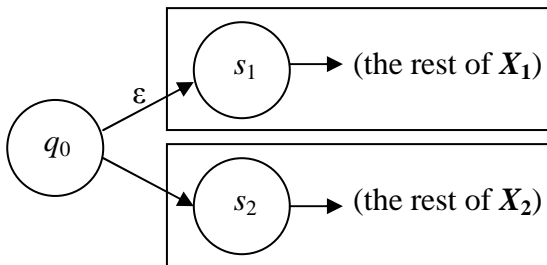
Consequence: a language is regular iff some NFA models it.

Why is this important? For one thing, NFAs are easier to construct and easier to read than DFAs. For instance:

- Let $X_1 = \langle Q_1, \Sigma, \delta_1, s_1, F_1 \rangle$ and $X_2 = \langle Q_2, \Sigma, \delta_2, s_2, F_2 \rangle$, where $|Q_1| = 13$ and $|Q_2| = 17$.

Find $X_{1 \cup 2}$, the union of X_1 and X_2 .

If we're using DFAs, then $Q_{1 \cup 2} = Q_1 \times Q_2$, and $|Q_{1 \cup 2}| = |Q_1| \cdot |Q_2| = 13 \cdot 17 = 221$ states. But with a NFA, we can do it in $|Q_1| + |Q_2| + 1 = 31$ states:



...and we're done, without having to worry about what X_1 and X_2 look like.