Non-Deterministic Finite State Automata (NFA or NFSA)

Q: What does deterministic mean?

The path is fixed \(\Rightarrow\) no choice.

NFSA. A non-deterministic finite state automata (NFSA) extends DFSA by allowing choice at each state.

Differences between DFSA and NFSA:

- **NFSA.** Given a state \(q_i\) and an input \(x\) there can be more than one possible transition, i.e,

\[
\delta^*(q, x) = \{\text{set of } q_i\}
\]

- **NFSA.** Given state \(q_i\), we can have an \(\epsilon\) transition.

\[
\delta^*(q_i, \epsilon) = q_j
\]

This means we can spontaneously jump from \(q_i\) to \(q_j\).

Q: How do we know if a string is accepted by an NFSA?

Check all possible paths and as long as one is accepted the string is accepted.
Examples of NFSA

Consider the strings that are *represented* by the *regular expression*: \((010 + 01)^*\)

**NFSA:**

[Diagram of NFSA]

**DFSA:**

[Diagram of DFSA]

**Formally.** An *NFSA*, is a machine \(M = (Q, \Sigma, \delta, s, F)\) where

- each of \(Q, \Sigma, s, F\) are as for a *DFSA*.

- \(\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)\) (*\(\mathcal{P}(Q)\) is a set of states).

- \(\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)\).
Limitations of DFSA and NFSA

Q: Can every set of strings be recognized by a DFSA? an NFSA?
   No.

Q: How much more powerful is an NFSA over a DFSA?
   Not at all.

Detailed answers.

- Only strings representable by regular expressions can be recognized by an NFSA or DFSA.
- There exists an algorithm to convert between deterministic and non-deterministic machines.

Theorem. If $L$ is a regular language then the following are all equivalent:

1. $L$ is denoted by a regular expression
2. $L$ is accepted by a deterministic FSA
3. $L$ is accepted by a non-deterministic FSA

(See the course text for the proof.)
Closure Properties of FSA-accepted Languages

Q: What do we mean by closure?

If we perform an operation on members of a set, the new elements belong to the set.

Theorem Every regular language \( L \) is closed under complementation, union, intersection, concatenation and the Kleene star operation.

Q: What does this mean?

If \( L, L' \) are regular then so are \( \overline{L}, L \cup L', L \cap L' \) and \( LL' \) and \( L^* \).

Proof of \( L \cup L' \).

- Let \( M \) be a NFSA that accepts \( L \).
- Let \( M' \) be a NFSA that accepts \( L' \).

Q: How can we construct \( M \cup M' \) that will accept either language?

Add a new start state with \( \epsilon \) transition to the start states of \( M, M' \).

Proof of \( L^* \).

Given \( M \) accepting \( L \), how can we build a new NFSA to accept \( L^* \)?

From each accepting state add an \( \epsilon \) transition to the start state and create a new start state to accept the \( \epsilon \) string.
Regular Languages

Q: How can we prove that a language $L$ is regular?

Q: How can we prove that a $L$ is not regular?

- Any FSA has a finite number of states, say $n$.

- Therefore if $L$ is infinite, then $L$ has strings with $> n$ symbols.

Q: What does this imply about at least one state of the FSA? Is visited more than once which creates a cycle.

- Repeating this cycle an arbitrary number of times must yield another string in $L$.

Q: What does this mean?

On a string with $> n$ symbols, there must be a portion that is just a cycle.

Q: How does this help us?

If we can create a string that is accepted but not in the language we have a contradiction.