Introduction

- Previous model is deterministic
  - Given a state and input, there is a unique state that we transition to next
- Nondeterminism is a generalization of determinism
  - Several choices may exist for what state to transition to next
- Every deterministic machine is a nondeterministic machine
- Does nondeterminism add more power?

Overview

- Equivalence of NFA and DFA
- Formal Definition
- Non-determinism
• How does it compute?
  - Any time there is a choice, run a copy of the machine for each choice
  - If no subsequent state, that copy (computation path) dies
  - If any computation path accepts, accept the string
  - If no subsequent state, then copy (computation path) dies
  - Any time there is a choice, run a copy of the machine for each choice

Computation

Example

Additions

+ Allow no input to transition
- Allow epsilon transitions
- Allow no transition from a state and token
- Allow multiple transitions from the same state and token
Example (continued)

What language does it accept?
- If guess is wrong, computation path dies out
- Non-determinism used to guess when end of string will be
- All state q1. If guess of 3 characters from the end
- When language does it accept?
Overview

- Non-determinism
- Formal Definition
- Equivalence of NFA and DFA

Another Example

- How do we modify the DFA?
- What language will it now accept?
- Add epsilon labels from $q_2$ to $q_3$ and from $q_3$ to $q_4$.

- Another Example
What is its formal definition?

Formal Definition

A non-deterministic finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:

1. $Q$ is a finite set of states
2. $\Sigma$ is a finite alphabet
3. $\delta : Q \times \Sigma \cup \{\epsilon\} \rightarrow P(Q)$ is the transition function
4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accept states

Note: $P(Q)$ is the power set of $Q$ — all possible subsets of $Q$.

Example
Overview

• Non-determinism

⇒ Equivalence of NFA and DFA

- Formal Definition

- Non-determinism

Definition of Computation

Definition:

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and $w$ be a string over $\Sigma$. We say that $N$ accepts $w$, if there is a sequence of states $q_0, q_1, \ldots, q_m$, such that:

1. $q_0 = q_0$
2. For $1 \leq i \leq m$, $q_i \in \delta(q_{i-1}, w_i)$
3. $q_m \in F$

It accepts if any legal sequence ends with an accept state.

Note that there are many choices for a sequence.

$\exists \delta \in \delta, q \in Q, \Sigma \subseteq \delta$ such that $w_0, \ldots, w_m \in \Sigma^*$, $w_0 = \varepsilon$, $w_i \neq \varepsilon$, and $w = w_0 w_1 \cdots w_m$.

Definition: Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and $w$ be a string over $\Sigma$. We say that $N$ accepts $w$, if there is a sequence of states $q_0, q_1, \ldots, q_m$ such that:

1. $q_0 = q_0$
2. For $1 \leq i \leq m$, $q_i \in \delta(q_{i-1}, w_i)$
3. $q_m \in F$

$\exists \delta \in \delta, q \in Q, \Sigma \subseteq \delta$ such that $w_0, \ldots, w_m \in \Sigma^*$, $w_0 = \varepsilon$, $w_i \neq \varepsilon$, and $w = w_0 w_1 \cdots w_m$.
Theorem 1.39: Every NFA has an equivalent DFA.

2 machines are equivalent if they recognize the same language.

Proof Idea (ignoring epsilon transitions):
- Construct DFA in which all possible subsets of states of the NFA are states of the DFA.
- If NFA has k states, DFA has 2^k states.
- Construct DFA in which all possible subsets of states of the NFA are states of the DFA.
- + Might be remove subset if computation path dead-ends, or converge.
- + Might add more fingers for splits.
- + To consume next input token update each finger.
- + Think of this as pointing a finger on each possible state that the NFA can be in.
- + As it is consuming its input it can be in one of a number of states.
- + DFA is basically exploring a bunch of paths in parallel.
- NFA is basically exploring a bunch of paths in parallel.

Theorem 1.39: Every NFA has an equivalent DFA.

2 machines are equivalent if they recognize the same language.

Use NFAs to show that regular languages are closed under composition.

We can use non-determinism to show that a language is regular.

NFA seems to be more powerful than DFA. The to non-determinism.

Surprising: if a language can be recognized by a DFA, it can be recognized by a NFA.

DFA and NFA recognize the same class of languages.

Equivalence of NFA and DFA.
**Proof (without Epsilon)**

Let $N = (Q, \Sigma, \delta, q_0, F)$ be a NFA that recognizes some language $A$.

Construct DFA $M = (Q', \Sigma, \delta', q'_0, F')$ recognizing $A$ as follows:

1. $Q' = \mathcal{P}(Q)$, where $\mathcal{P}(Q)$ is the powerset of $Q$ — all subsets of $Q$.

2. For $R \in Q'$ and $a \in \Sigma$, let $\delta'(R, a) = \{q | q \in \delta(r, a) \text{ for some } r \in R\}$ for some $r \in R$, and $a \in \Sigma$.

3. $q'_0 = \{q_0\}$.

4. $F' = \{R \in Q' | R \text{ contains an accept state of } N\}$.

$M$ accepts if one of the possible states that $N$ could be in at this point is an accept state.

Need to argue that for any string $w$, $M$ accepts $w$ iff $N$ accepts $w$.

At each point in the computation, the possible states that $N$ can be in is the meta-state that $M$ is in.

Would need to do this by induction. Simple enough that we won't bother.

---

**Example**

Let an NFA that recognizes language where 2nd last token is a 1.
Example

NFA that recognizes language where last or 2nd last token is 1

- Consider $R \subseteq Q^+$

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

  Modify the transition function of $M$ to take into account the states in $N$ reachable after epsilon transitions.

- $\delta'(R, a) = \{ q \mid q \in \delta'(r, a) \text{ for some } r \in R \}$

- Also need to modify start state of $M$ to be set of states reachable from $

\{ h \}

\epsilon = \emptyset

\text{after every step (this will be simpler than before each step)}$

As we are treating our computation in $N$, place additional integers
Corollary 1.40: A language is regular iff some NFA recognizes it.

Proof:

⇒
Let \( L \) be a regular language.
By last theorem, there is a DFA that recognizes the language.

Let \( M \) be a DFA that recognizes \( L \).
Then \( M \) is also a NFA.

⇐
Let \( N \) be a NFA that recognizes \( L \).
By last theorem, there is a DFA that recognizes the language.
So, \( N \) is a regular language.