Automata Theory and Formal Grammars: Lecture 2

Deterministic and Nondeterministic Finite Automata

Deterministic and Nondeterministic Finite Automata

Last Time

- Sets Theory (Review?)
- Logic, Proofs (Review?)
- Words, and operations on them: $w_1 \circ w_2, w^i, w^*, w^+$
- Languages, and operations on them: $L_1 \circ L_2, L^i, L^*, L^+$

Today

- Deterministic Finite Automata (DFAs) and their languages
- Closure properties of DFA languages (the product construction)
- Nondeterministic Finite Automata (NFAs) and their languages
- Relating DFAs and NFAs (the subset construction)

Fibonacci as a Recursively Defined Set

The n^{th} Fibonacci number $f(n)$:

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 1 \\ f(n) &= f(n-1) + f(n-2), \text{ for } n \geq 2 \end{aligned}$$

As a recursively defined set (relation)

$$\begin{aligned} F_0 &= \emptyset \\ F_{i+1} &= \{\langle 0,0\rangle, \langle 1,1\rangle\} \\ &\cup \left\{ \langle n, f_{n_1} + f_{n_2}\rangle \;\middle|\; \begin{array}{l} \langle n_1, f_{n_1}\rangle \in F_i \text{ and} \\ \langle n_2, f_{n_2}\rangle \in F_i \text{ and} \\ n = n_1 + 1 = n_2 + 2 \end{array} \right\} \end{aligned}$$

Fibonacci as a Recursively Defined Set

$$\begin{aligned} F_0 &= \emptyset \\ F_{i+1} &= \{\langle 0,0\rangle, \langle 1,1\rangle\} \\ &\cup \left\{ \langle n, f_{n_1} + f_{n_2}\rangle \;\middle|\; \begin{array}{l} \langle n_1, f_{n_1}\rangle \in F_i \text{ and} \\ \langle n_2, f_{n_2}\rangle \in F_i \text{ and} \\ n = n_1 + 1 = n_2 + 2 \end{array} \right\} \end{aligned}$$

For example:

$$\begin{aligned} F_0 &= \emptyset \\ F_1 &= \{\langle 0,0\rangle, \langle 1,1\rangle\} \\ F_2 &= \{\langle 0,0\rangle, \langle 1,1\rangle, \langle 2,1\rangle\} \\ F_3 &= \{\langle 0,0\rangle, \langle 1,1\rangle, \langle 2,1\rangle, \langle 3,2\rangle\} \\ F_4 &= \{\langle 0,0\rangle, \langle 1,1\rangle, \langle 2,1\rangle, \langle 3,2\rangle, \langle 4,3\rangle\} \\ F_5 &= \end{aligned}$$

Conventions

- Σ is an arbitrary alphabet. (In examples, Σ should be clear from context.)
- The variables a–e range over letters in Σ.
- The variables u–z range over words over Σ^*.
- The variables p–q range over states in Q.

Recall

For any string w and language L:

$$w \circ \varepsilon = w \qquad = \varepsilon \circ w \tag{1}$$

$$L \circ \{\varepsilon\} = L \qquad = \{\varepsilon\} \circ L \tag{2}$$

$$L^* = \{\varepsilon\} \cup L \circ L^* \tag{3}$$

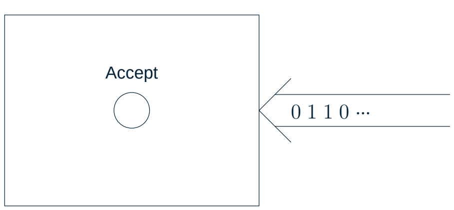
L^* is closed with respect to concatenation, for any L:

if $u \in L^*$ and $v \in L^*$ then $u \circ v \in L^*$

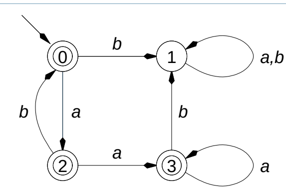
Finite Automata

... are "machines" for recognizing languages!

- They process input words a symbol at a time.
- An "accept light" flashes if the symbols read in so far are "OK".



Formal Definition of Finite Automata



$\boxed{\text{Definition}}$ A finite automaton (DFA) is a quintuple $\langle Q, \Sigma, q_0, \delta, A\rangle$ where:

- Q is a finite non-empty set of states;
- Σ is an alphabet;
- $q_0 \in Q$ is the start state;
- $\delta : Q \times \Sigma \to Q$ is the transition function; and
- $A \subseteq Q$ is the set of accepting (final) states.

DFA Acceptance

Given a DFA $M = \langle Q, \Sigma, q_0, \delta, A \rangle$ and word $w \in \Sigma^*$:

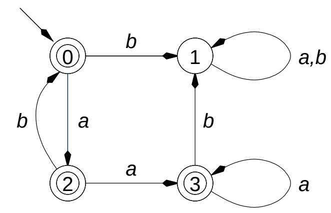
- M should accept w if in processing w a symbol at a time, M goes to an accepting state.
- To formalize this we define a function

$$\delta^* : Q \times \Sigma^* \to Q$$

$\delta^*(q, w)$ should be the state reached from q after processing w.

- How to define δ^*?

Example of δ^*



$$\begin{aligned} \delta^*(0, aab) &= \delta^*(\delta(0,a), ab) = \delta^*(2, ab) \\ &= \delta^*(\delta(2,a), b) = \delta^*(3, b) \\ &= \delta^*(\delta(3,b), \varepsilon) = \delta^*(1, \varepsilon) \\ &= 1 \end{aligned}$$

What is $\delta^*(0, abaa)$?

Definition of δ^*

Definition Let $M = \langle Q, \Sigma, q_0, \delta, A \rangle$ be a DFA. Then $\delta^* : Q \times \Sigma^* \to Q$ is defined recursively:

$$\delta^*(q, w) = \begin{cases} q & \text{if } w = \varepsilon \\ \delta^*(\delta(q,a), w') & \text{if } w = aw' \text{ and } a \in \Sigma \end{cases}$$

$\delta^*(q, w) = q'$ if q' the state reached by processing w, starting from q.

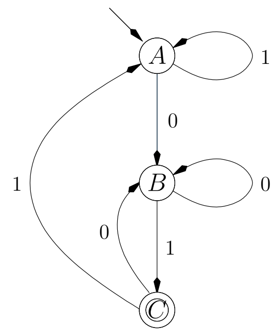
Language of a Finite Automaton

A DFA accepts a word if it reaches an accepting state after “consuming” the word.

Definition Let $M = \langle Q, \Sigma, q_0, \delta, A \rangle$ be a DFA.

- M accepts $w \in \Sigma^*$ if $\delta^*(q_0, w) \in A$.
- $\mathcal{L}(M) = \{\, w \in \Sigma^* \mid M \text{ accepts } w \,\}$ is the language accepted by M.

Example: DFA for $\{ w \in \{0,1\}^* \mid w \text{ ends in } 01 \}$



Example: DFA for Valid Binary Numbers

- Must contain at least one digit.
- No leading 0s.

DFA Languages

Definition A language $L \subseteq \Sigma^*$ is a DFA language if there exists a DFA M such that $L = \mathcal{L}(M)$.

- Is the set of Java numeric constants a DFA language?

  `0xE`, `15`, `017`, `15l`, `15L`, `15.0`, `1.5e1`, `1.5E1`

  Yes! To show it build a DFA.

- Is the set of strings of balanced parentheses a DFA language?

  ε, `ab`, `aabb`, `aaabbb`,

  No! To show it . . . attend lecture 4.

Closure Properties for DFA Languages

Closed Sets

Let f be a unary operation $f : U \to U$. A subset $S \subseteq U$ is closed under f — equivalently, f preserves S — if

$$\forall s \in S.\ f(s) \in S$$

Let g be a binary operation $g : U \times U \to U$. A subset $S \subseteq U$ is closed under g — equivalently, g preserves S — if

$$\forall \langle s_1, s_2 \rangle \in S \times S.\ f(s_1, s_2) \in S$$

- Naturals are closed under addition, not subtraction.
- Integers are closed under multiplication, not division.
- Rationals are closed under division, not square root.
- Reals are closed under square root, not exponentiation.
- Complex are closed under exponentiation.

Closure Properties for DFA Languages

- We would like to see what operations on languages “preserve” the property of being recognizable by a DFA.
- For example, suppose we wish to show the following:

  Let L_1 and L_2 be DFA languages. Then $\overline{L_1}$ and $L_1 \cap L_2$ are also DFA languages.
- How do we prove this? Via constructions.

Complementation

$\boxed{\text{Theorem}}$ Let $L \subseteq \Sigma^*$ be a DFA language. Then so is $\overline{L}$.

Since L is a DFA language we know there is a DFA M accepting it. How can we build a DFA for $\overline{L}$?

$\boxed{\text{Idea}}$ Reverse the accepting and nonaccepting states in M!

The proof formalizes this idea.

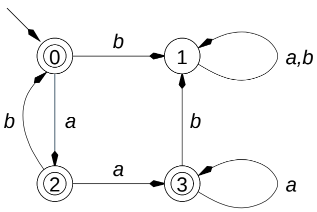
Proof

- Suppose L is a DFA language. By definition, there must be a DFA M such that $\mathcal{L}(M) = L$
- Fix $M = \langle Q, \Sigma, q_0, \delta, A \rangle$.
- Let $\overline{M} = \langle Q, \Sigma, q_0, \delta, Q - A \rangle$. We show that $\mathcal{L}(\overline{M}) = \overline{L}$.
  - For any $w \in \Sigma^*$,

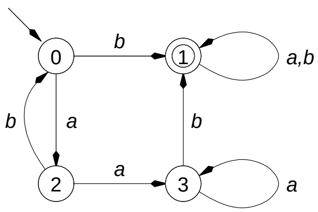
    $$\delta^*(q_0, w) \notin A \quad \text{iff} \quad \delta^*(q_0, w) \in Q - A$$

    This holds trivially by induction on length of w.
  - Thus, for any w, $w \in \mathcal{L}(\overline{M})$ if and only if $w \notin \mathcal{L}(M)$.
  - Thus, $\mathcal{L}(\overline{M}) = \overline{L}$. QED

Example of Complementation Construction



M



M'

Intersection

Theorem Let $L_1, L_2 \subseteq \Sigma^*$ be DFA languages. Then $L_1 \cap L_2$ is a DFA language.

To prove this we will use the Product Construction.

- Given two DFAs M and N, the product construction builds a new DFA $\Pi(M, N)$ that "runs" M and N in parallel.
- $\Pi(M, N)$ then accepts a word iff both M and N do.
- So $\mathcal{L}(\Pi(M, N)) = \mathcal{L}(M) \cap \mathcal{L}(N)$!

How do we define Π?

The Product Construction

Let $M = \langle Q_M, \Sigma, q_M, \delta_M, A_M \rangle$ be a DFA.
Let $N = \langle Q_N, \Sigma, q_N, \delta_N, A_N \rangle$ be a DFA.
Define $\Pi(M, N)$ as

$$\Pi(M, N) = \langle Q_M \times Q_N \, , \, \Sigma \, , \, \langle q_M, q_N \rangle \, , \, \delta_{MN} \, , \, A_M \times A_N \rangle$$

where

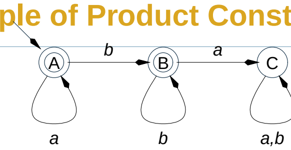
$$\delta_{MN}(\langle q_1, q_2 \rangle, a) = \langle \delta_M(q_1, a), \delta_N(q_2, a) \rangle$$

Lemma For any $w \in \Sigma^*, q_1 \in Q_M$, and $q_2 \in Q_N$,

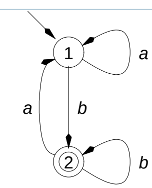
$$\delta^*_{MN}(\langle q_1, q_2 \rangle, w) = \langle \delta^*_M(q_1, w), \delta^*_N(q_2, w) \rangle.$$

Proof?
And how does this help show that $\mathcal{L}(\Pi(M, N)) = \mathcal{L}(M) \cap \mathcal{L}(N)$?

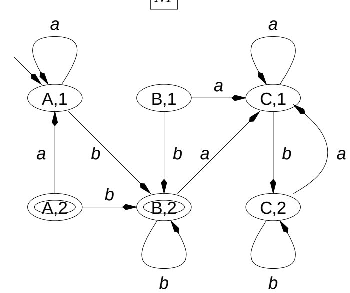
Example of Product Construction



$\boxed{M}$



$\boxed{N}$



$\boxed{\Pi(M, N)}$

A Corollary about Closure for DFA Languages

What's a "corollary"? An "obvious consequence".

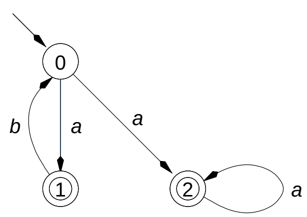
Corollary Let $L_1, L_2 \subseteq \Sigma^*$ be DFA languages. Then so are $L_1 \cup L_2$ and $L_1 - L_2$.

Why is this an "obvious consequence" of what we have seen before?

Nondeterministic Finite Automata

Nondeterministic Finite Automata

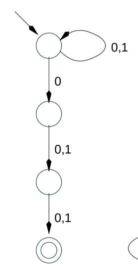
- Regular DFAs require exactly one transition per state for each input symbol.
- Nondeterministic FAs allow any number of transitions!



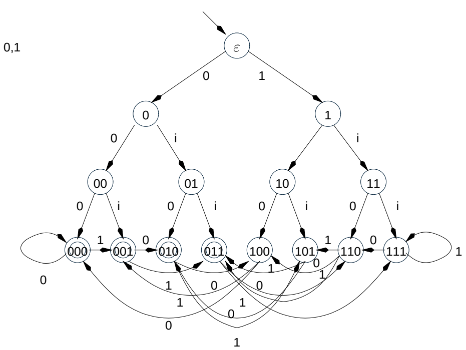
Why study NFAs? Because they are easier to work with sometimes....

NFAs Can Be Smaller Than DFAs!

Consider language $L \subseteq \{0,1\}^*$ given by regular expression $(0+1)^*0(0+1)(0+1)$ (3^{rd} symbol from right is a 0).



NFA



FA

Formal Definition of NFAs

$\boxed{\text{Definition}}$ A nondeterministic finite automaton (NFA) is a quintuple $\langle Q, \Sigma, q_0, \delta, A \rangle$ where:

- Q is a finite set of states;
- Σ is the input alpabet;
- $q_0 \in Q$ is the start state;
- $A \subseteq Q$ is the set of accepting states; and
- $\delta : Q \times \Sigma \to 2^Q$ is the transition function.

$\boxed{\text{Idea}}$ $\delta(q, a)$ records the set of states reachable from q via an a-transition.

Languages of NFAs: Defining δ^*

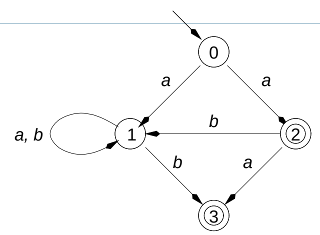
To formalize acceptance we first define a function $\delta^* : Q \times \Sigma^* \to 2^Q$; $\delta^*(q, w)$ contains all the states reachable from q after processing w.

$\boxed{\text{Definition}}$ Let $M = \langle Q, \Sigma, q_0, \delta, A \rangle$ be a NFA. Then $\delta^* : Q \times \Sigma^* \to 2^Q$ is defined as follows.

$$\delta^*(q, w) = \begin{cases} \{q\} & \text{if } w = \varepsilon \\ \bigcup_{q' \in \delta(q,a)} \delta^*(q', w') & \text{if } w = aw' \text{ and } a \in \Sigma \end{cases}$$

Note that $\delta^*(q, w)$ gives us the set of all possible outcomes of processing w from state q.

Example



$$\begin{aligned}
\delta^*(0, abb) &= \bigcup_{q' \in \delta(0,a)} \delta^*(q', bb) = \bigcup_{q' \in \{1,2\}} \delta^*(q', bb) \\
&= \delta^*(1, bb) \cup \delta^*(2, bb) \\
&= \bigcup_{q' \in \delta(1,b)} \delta^*(q', b) \cup \bigcup_{q' \in \delta(2,b)} \delta^*(q', b) \\
&= \delta^*(1, b) \cup \delta^*(3, b) \cup \delta^*(1, b) \\
&= \delta^*(1, \varepsilon) \cup \delta^*(3, \varepsilon) \cup \emptyset \\
&= \{1\} \cup \{3\} = \{1, 3\}
\end{aligned}$$

Languages of NFAs

- As with DFAs, the language of a NFA consists of the words that it accepts.
- In a NFA nondeterministic choices require “guessing”: which transition should be taken? Some paths may lead to accepting states, whereas others do not.
- A NFA accepts a word if it is possible to make the guesses so that we reach an accepting state.
- This is called angelic nondetermism. We have access to an oracle that always guesses correctly.

Languages of NFAs: Formal Definition

Definition Let $M = \langle Q, \Sigma, q_0, \delta, A \rangle$ be a NFA, and let $w \in \Sigma^*$.

- M accepts w if $\delta^*(q_0, w) \cap A \neq \emptyset$.
- The language, $\mathcal{L}(M)$, of M is defined by:
  $\mathcal{L}(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$

So M accepts w if it is possible to reach an accepting state after processing w.

So What Is Relationship Between DFAs and NFAs?

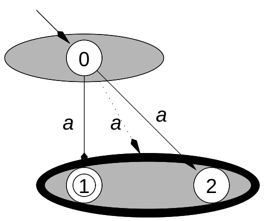
Theorem

1. For any DFA M there is a NFA N such that $\mathcal{L}(N) = \mathcal{L}(M)$.
2. For any NFA N there is a DFA M such that $\mathcal{L}(M) = \mathcal{L}(N)$.

Proof of 1 is easy, since any DFA “is” a NFA. But 2?

- Idea behind proof is to define DFA that “tracks” behavior of NFA on a given input word.
- This construction is often called the subset construction because states in the DFA correspond to set of states in the NFA.

The Subset Construction: Intuition



The Subset Construction

Let $N = \langle Q, \Sigma, q_0, \delta, A \rangle$ be a NFA.
We want to construct a DFA $D(N)$ accepting the same language.
States in $D(N)$ will be sets of states from N.
Let P range over states of $D(N)$.
$P \in 2^Q$, that is, $P \subseteq Q$.

$$D(N) = \langle 2^Q \,,\, \Sigma \,,\, \{q_0\} \,,\, \delta_{DN} \,,\, A_{DN} \rangle$$

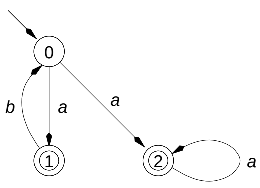
where

$$\delta_{DN}(P, a) = \bigcup_{q \in P} \delta(q, a)$$

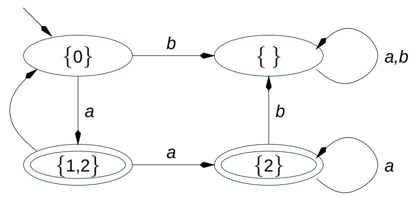
$$A_{DN} = \{ P \mid P \in 2^Q \text{ and } P \cap A \neq \emptyset \}$$

Note that $\delta^*(q_0, w) \in Q$, whereas $\delta^*_{DN}(\{q_0\}, w) \subseteq Q$.

Example of Subset Construction



N



D(N)

Note. Only reachable states in $D(N)$ are represented. (In practice, not all subsets of Q are reachable from $\{q_0\}$, and these need not be added explicitly to $D(N)$.)

Correctness of Subset Construction

Let $N = \langle Q, \Sigma, q_0, \delta, A\rangle$ be a NFA.

$D(N) = \langle 2^Q \,,\, \Sigma \,,\, \{q_0\} \,,\, \delta_{DN} \,,\, A_{DN}\rangle$ where — for $P \in 2^Q$ — $\delta_{DN}(P, a) = \bigcup_{q \in P} \delta(q, a)$ and $A_{DN} = \{\, P \mid P \cap A \neq \emptyset \,\}$.

Theorem For any NFA N, $\mathcal{L}(N) = \mathcal{L}(D(N))$.

Recall that $\delta^*(q_0, w) \in Q$, whereas $\delta^*_{DN}(\{q_0\}, w) \subseteq Q$.

The proof relies on the following observations. For any $w \in \Sigma^*$:

- $\delta^*(q_0, w) = \delta^*_{DN}(\{q_0\}, w)$
- $\delta^*(q_0, w) \cap A \neq \emptyset$ if and only if $\delta^*_{DN}(\{q_0\}, w) \in A_{DN}$

Consequently, $w \in \mathcal{L}(N)$ if and only if $w \in \mathcal{L}(D(N))$!