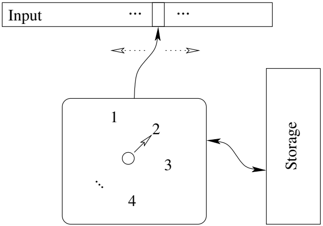
CS 455/555: Finite automata

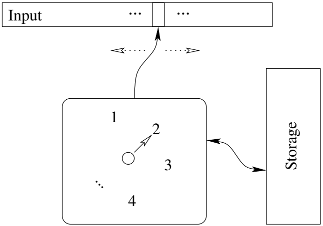
Stefan D. Bruda

Fall 2020

- Generally any automaton
  - Has a finite-state control
  - Scans the input one symbol at a time
  - Takes an action based on the currently scanned symbol and the current state
  - The action taken may yield a different current state
  - May make use of some form of extra storage



- Generally any automaton
  - Has a finite-state control
  - Scans the input one symbol at a time
  - Takes an action based on the currently scanned symbol and the current state
  - The action taken may yield a different current state
  - May make use of some form of extra storage



- A finite automaton scans the input from left to right only and uses no additional storage
  - It cannot go back in the input
  - It can only remember (using the finite state control) a finite amount of information about the already seen input

- A deterministic finite automaton is a tuple $M = (K, \Sigma, \delta, s, F)$
  - $K \Rightarrow$ finite set of states
  - $\Sigma \Rightarrow$ input alphabet
  - $F \subseteq K \Rightarrow$ set of final states
  - $s \in K \Rightarrow$ initial state
  - $\delta : K \times \Sigma \rightarrow K \Rightarrow$ transition function
- Configuration: $c \in K \times \Sigma^*$
- Yields in one step: $(q, aw) \vdash_M (q', w)$ iff $a \in \Sigma$ and $\delta(q, a) = q'$
  - $\vdash_M^* \Rightarrow$ reflexive and transitive closure
- w is accepted by M iff $\exists\, q \in F : (s, w) \vdash_M^* (q, \varepsilon)$
- The language accepted by M:

$$\mathcal{L}(M) = \{w \in \Sigma^* : \exists\, q \in F : (s, w) \vdash_M^* (q, \varepsilon)\}$$

- A nondeterministic finite automaton is a tuple $M = (K, \Sigma, \Delta, s, F)$
  - $K \Rightarrow$ finite set of states
  - $\Sigma \Rightarrow$ input alphabet
  - $F \subseteq K \Rightarrow$ set of final states
  - $s \in K \Rightarrow$ initial state
  - $\Delta \subseteq K \times (\Sigma \cup \{\varepsilon\}) \times K \Rightarrow$ transition relation
- Configuration: $c \in K \times \Sigma^*$
- Yields in one step: $(q, aw) \vdash_M (q', w)$ iff $a \in \Sigma \cup \{\varepsilon\}$ and $(q, a, q') \in \Delta$
  - $\vdash_M^* \Rightarrow$ reflexive and transitive closure
- w is accepted by M iff $\exists\, q \in F : (s, w) \vdash_M^* (q, \varepsilon)$
- The language accepted by M:

$$\mathcal{L}(M) = \{w \in \Sigma^* : \exists\, q \in F : (s, w) \vdash_M^* (q, \varepsilon)\}$$

- Languages accepted by finite automata?
  - Of finite strings for sure
- Deterministic FA $\Rightarrow$special case of nondeterministic FA
- In fact the two kind of finite automaton accept the same languages
- $M = (K, \Sigma, \Delta, s, F) \Rightarrow M' = (K', \Sigma, \delta', s', F')$ such that $\mathcal{L}(M) = \mathcal{L}(M')$
  - $K' = 2^K$
  - Let $E(q)$ be the closure of $\{q\}$ under $\{(p, r) : (p, \varepsilon, r) \in \Delta\}$
  - $s' = E(s)$
  - $F' = \{Q \subseteq K : Q \cap F \neq \emptyset\}$
  - $\delta'(Q, a) = \bigcup\{E(p) : p \in K, (q, a, p) \in \Delta \text{ for some } q \in Q\}$

  (proof on p. 71)
- DFA are more efficient, potentially difficult to understand, and often considerably larger (how much larger?)

- $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ and $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$. Can we construct $M = (K, \Sigma, \Delta, s, F)$ such that
  - $\mathcal{L}(M) = \mathcal{L}(M_1) \cup \mathcal{L}(M_2)$ (closure under union)?
  - $\mathcal{L}(M) = \mathcal{L}(M_1)\mathcal{L}(M_2)$ (closure under concatenation)?
  - $\mathcal{L}(M) = \mathcal{L}(M_1)^*$ (closure under Kleene star)?
  - $\mathcal{L}(M) = \overline{\mathcal{L}(M_1)}$ (closure under complement)?
  - $\mathcal{L}(M) = \mathcal{L}(M_1) \cap \mathcal{L}(M_2)$ (closure under intersection)?

- $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ and $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$. Can we construct $M = (K, \Sigma, \Delta, s, F)$ such that
  - $\mathcal{L}(M) = \mathcal{L}(M_1) \cup \mathcal{L}(M_2)$ (closure under union)?
    $K = K_1 \cup K_2 \qquad F = F_1 \cup F_2 \qquad \Delta = \Delta_1 \cup \Delta_2 \cup \{(s, \varepsilon, s_1), (s, \varepsilon, s_2)\}$
  - $\mathcal{L}(M) = \mathcal{L}(M_1)\mathcal{L}(M_2)$ (closure under concatenation)?
  - $\mathcal{L}(M) = \mathcal{L}(M_1)^*$ (closure under Kleene star)?
  - $\mathcal{L}(M) = \overline{\mathcal{L}(M_1)}$ (closure under complement)?
  - $\mathcal{L}(M) = \mathcal{L}(M_1) \cap \mathcal{L}(M_2)$ (closure under intersection)?

Closure Properties

- $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ and $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$. Can we construct $M = (K, \Sigma, \Delta, s, F)$ such that
  - $\mathcal{L}(M) = \mathcal{L}(M_1) \cup \mathcal{L}(M_2)$ (closure under union)?
    $K = K_1 \cup K_2 \qquad F = F_1 \cup F_2 \qquad \Delta = \Delta_1 \cup \Delta_2 \cup \{(s, \varepsilon, s_1), (s, \varepsilon, s_2)\}$
  - $\mathcal{L}(M) = \mathcal{L}(M_1)\mathcal{L}(M_2)$ (closure under concatenation)?
    $s = s_1 \qquad F = F_2 \qquad \Delta = \Delta_1 \cup \Delta_2 \cup \{(f, \varepsilon, s_2) : f \in F_1\}$
  - $\mathcal{L}(M) = \mathcal{L}(M_1)^*$ (closure under Kleene star)?
    $s = s_1 \qquad F = F_1 \qquad \Delta = \Delta_1 \cup \{(f, \varepsilon, s_1) : f \in F_1\} \cup \{(s_1, \varepsilon, f) : f \in F_1\}$
  - $\mathcal{L}(M) = \overline{\mathcal{L}(M_1)}$ (closure under complement)?
    $s = s_1 \qquad \delta = \delta_1 \qquad F_1 = K \setminus F$
  - $\mathcal{L}(M) = \mathcal{L}(M_1) \cap \mathcal{L}(M_2)$ (closure under intersection)?

- $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ and $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$. Can we construct $M = (K, \Sigma, \Delta, s, F)$ such that
  - $\mathcal{L}(M) = \mathcal{L}(M_1) \cup \mathcal{L}(M_2)$ (closure under union)?
    $K = K_1 \cup K_2 \qquad F = F_1 \cup F_2 \qquad \Delta = \Delta_1 \cup \Delta_2 \cup \{(s, \varepsilon, s_1), (s, \varepsilon, s_2)\}$
  - $\mathcal{L}(M) = \mathcal{L}(M_1)\mathcal{L}(M_2)$ (closure under concatenation)?
    $s = s_1 \qquad F = F_2 \qquad \Delta = \Delta_1 \cup \Delta_2 \cup \{(f, \varepsilon, s_2) : f \in F_1\}$
  - $\mathcal{L}(M) = \mathcal{L}(M_1)^*$ (closure under Kleene star)?
    $s = s_1 \qquad F = F_1 \qquad \Delta = \Delta_1 \cup \{(f, \varepsilon, s_1) : f \in F_1\} \cup \{(s_1, \varepsilon, f) : f \in F_1\}$
  - $\mathcal{L}(M) = \overline{\mathcal{L}(M_1)}$ (closure under complement)?
    $s = s_1 \qquad \delta = \delta_1 \qquad F_1 = K \setminus F$
  - $\mathcal{L}(M) = \mathcal{L}(M_1) \cap \mathcal{L}(M_2)$ (closure under intersection)?
    $\mathcal{L}(M_1) \cap \mathcal{L}(M_2) = \overline{\overline{\mathcal{L}(M_1)} \cup \overline{\mathcal{L}(M_1)}}$

Closure under Intersection (constructive)

- $M_1 = (K_1, \Sigma, s_1, \delta_1, F_1)$, $M_2 = (K_2, \Sigma, s_2, \delta_2, F_2) \Rightarrow M = (K, \Sigma, s, \delta, F)$ such that $\mathcal{L}(M_1) \cap \mathcal{L}(M_2) = \mathcal{L}(M)$
- M must somehow run M_1 and M_2 "in parallel" to determine whether both accept the input
- It follows that at any given time we have to keep track of the current states of both M_1 and M_2. We thus put $K = K_1 \times K_2$
- At the beginning of the computation both M_1 and M_2 are in their respective initial states, so $s = (s_1, s_2)$
- Similarly, in order for the input to be accepted, both M_1 and M_2 must be in one of their respective final states, so $F = F_1 \times F_2$
- Finally, δ should allow M to perform simultaneously exactly one transition of M_1 and exactly one transition of M_2: $\delta((q_1, q_2), a) = (q'_1, q'_2)$ iff $\delta_1(q_1, a) = q'_1$ and $\delta_2(q_2, a) = q'_2$

- **Theorem:** Finite automata accept exactly all the languages in REG

- **Theorem:** Finite automata accept exactly all the languages in REG
- $\supseteq$:
  - REG = closure of $\{\{a\} : a \in \Sigma\} \cup \emptyset$ under union, concatenation, and Kleene star
  - Clearly FA accept $\{a\}$, $\emptyset$ and are closed under the above operations
  - So FA accept all REG (closure is minimal)

- **Theorem:** Finite automata accept exactly all the languages in REG
- $\supseteq$:
  - REG = closure of $\{\{a\} : a \in \Sigma\} \cup \emptyset$ under union, concatenation, and Kleene star
  - Clearly FA accept $\{a\}$, $\emptyset$ and are closed under the above operations
  - So FA accept all REG (closure is minimal)
- $\subseteq$:
  - Let $M = (\{q_1, q_2, \ldots, q_n\}, \Sigma, \Delta, q_1, F)$
  - Let $\langle i, j, k\rangle$ be the path from q_i to q_j of rank k (i.e., q_α in the path implies $\alpha \leq k$)
  - Let $R(i, j, k)$ be the set of strings in Σ^* along all the paths $\langle i, j, k\rangle$
  - Obviously, $\mathcal{L}(M) = \bigcup\{R(1, j, n) : q_j \in F\}$
  - We prove that all $R(i, j, k)$ are regular

Languages Accepted by Finite Automata

- **Theorem:** Finite automata accept exactly all the languages in REG
- $\supseteq$:
  - REG = closure of $\{\{a\} : a \in \Sigma\} \cup \emptyset$ under union, concatenation, and Kleene star
  - Clearly FA accept $\{a\}$, $\emptyset$ and are closed under the above operations
  - So FA accept all REG (closure is minimal)
- $\subseteq$:
  - Let $M = (\{q_1, q_2, \ldots, q_n\}, \Sigma, \Delta, q_1, F)$
  - Let $\langle i, j, k\rangle$ be the path from q_i to q_j of rank k (i.e., q_α in the path implies $\alpha \le k$)
  - Let $R(i, j, k)$ be the set of strings in Σ^* along all the paths $\langle i, j, k\rangle$
  - Obviously, $\mathcal{L}(M) = \bigcup\{R(1, j, n) : q_j \in F\}$
  - We prove that all $R(i, j, k)$ are regular by induction over k
  - basis: All the $\langle i, j, 0\rangle$ are transitions of M only, so $R(i, j, k)$ are clearly regular
  - inductive hypothesis: all the $R(i, j, k-1)$ are regular
  - $R(i, j, k) = R(i, j, k-1) \cup R(i, k, k-1)R(k, k, k-1)^*R(k, j, k-1)$
  - then $R(i, j, k)$ are regular given the closure of regular expressions under union, concatenation, and Kleene star

- Easy to eliminate unreachable states, but this does not yield an optimal automaton
- Can also merge states that are equivalent to others

- Easy to eliminate unreachable states, but this does not yield an optimal automaton
- Can also merge states that are equivalent to others
  - Equivalent states are states that produce the same strings
- Let $L \subseteq \Sigma^*$ and $x, y \in \Sigma^*$. Then $x \approx_L y$ if $xz \in L$ iff $yz \in L$ for all $x \in \Sigma^*$
  - $[x] = \{y \in \Sigma^* : y \approx_L x\}$

State minimization

- Easy to eliminate unreachable states, but this does not yield an optimal automaton
- Can also merge states that are equivalent to others
  - Equivalent states are states that produce the same strings
- Let $L \subseteq \Sigma^*$ and $x, y \in \Sigma^*$. Then $x \approx_L y$ if $xz \in L$ iff $yz \in L$ for all $x \in \Sigma^*$
  - $[x] = \{y \in \Sigma^* : y \approx_L x\}$
- Let $M = (K, \Sigma, \delta, s, f)$. Then $x \backsim_M y$ iff there exists $q \in K$ such that $(s, x) \vdash_M^* (q, \varepsilon)$ and $(s, y) \vdash_M^* (q, \varepsilon)$
  - $x \backsim_M y$ implies $x \approx_{\mathcal{L}(M)} y$
  - The number of states of M must be at least as large as the number of equivalence classes in $\mathcal{L}(M)$ under $\approx$

Theorem

Let $L \in \Sigma^*$ be a regular language. Then there exists a deterministic finite automaton with precisely as many states as there are equivalence classes in $\approx_L$

- Easy to eliminate unreachable states, but this does not yield an optimal automaton
- Can also merge states that are equivalent to others
    - Equivalent states are states that produce the same strings
- Let $L \subseteq \Sigma^*$ and $x, y \in \Sigma^*$. Then $x \approx_L y$ if $xz \in L$ iff $yz \in L$ for all $x \in \Sigma^*$
    - $[x] = \{y \in \Sigma^* : y \approx_L x\}$
- Let $M = (K, \Sigma, \delta, s, f)$. Then $x \backsim_M y$ iff there exists $q \in K$ such that $(s, x) \vdash_M^* (q, \varepsilon)$ and $(s, y) \vdash_M^* (q, \varepsilon)$
    - $x \backsim_M y$ implies $x \approx_{\mathcal{L}(M)} y$
    - The number of states of M must be at least as large as the number of equivalence classes in $\mathcal{L}(M)$ under $\approx$

Theorem

Let $L \in \Sigma^*$ be a regular language. Then there exists a deterministic finite automaton with precisely as many states as there are equivalence classes in $\approx_L$

$K = \{[x] : x \in \Sigma^*\}$, the set of equivalence classes under $\approx_L$

$s = [\varepsilon] \qquad F = \{[x] : x \in L\} \qquad \delta([x], a) = [xa]$

- Let $M = (K, \Sigma, \delta, s, f)$. Let $A_M \subseteq K \times \Sigma^*$ such that $(q, w) \in A_M$ iff $(q, w) \vdash_M^* (f, \varepsilon)$
- Let $q \equiv p$ iff for all $z \in \Sigma^*$: $(q, z) \in A_M$ iff $(p, z) \in A_M$
- $\equiv$ can be computed iteratively ($\equiv_0$, $\equiv_1$, $\equiv_2$, ...) as follows:
  1. $\equiv_0$ partitions K into F and $K \setminus F$
  2. **repeat** for $n \in \mathbb{N}$:
     1. $q \equiv_n p$ whenever $q \equiv_{n-1} p$ and $\delta(q, a) \equiv_{n-1} \delta(p, a)$ for all $a \in \Sigma$
  3. **until** $\equiv_n$ is the same as $\equiv_{n-1}$

ALGORITHM FOR STATE MINIMIZATION

- Let $M = (K, \Sigma, \delta, s, f)$. Let $A_M \subseteq K \times \Sigma^*$ such that $(q, w) \in A_M$ iff $(q, w) \vdash_M^* (f, \varepsilon)$
- Let $q \equiv p$ iff for all $z \in \Sigma^*$: $(q, z) \in A_M$ iff $(p, z) \in A_M$
- $\equiv$ can be computed iteratively ($\equiv_0$, $\equiv_1$, $\equiv_2$, ...) as follows:
  1. $\equiv_0$ partitions K into F and $K \setminus F$
  2. **repeat** for $n \in \mathbb{N}$:
     1. $q \equiv_n p$ whenever $q \equiv_{n-1} p$ and $\delta(q, a) \equiv_{n-1} \delta(p, a)$ for all $a \in \Sigma$
  3. **until** $\equiv_n$ is the same as $\equiv_{n-1}$
- $\equiv_n$ is a proper refinement of $\equiv_{n-1}$ so the algorithm terminates after at most $|K| - 1$ iterations $\Rightarrow$ polynomial complexity

- nondeterministic to deterministic FA $\Rightarrow$exponential time
- nondeterministic FA to regular expression $\Rightarrow$exponential time
  - $O(|K|)$ computations of $R(i, j, k)$, but $R(i, j, k)$ doubles each time
- Whether two FA or regular expressions accept/generate the same language
  - polynomial time for DFA
  - likely exponential time for NFA, regular expressions
- Decide whether $w \in \mathcal{L}(M)$:
  - $O(|w|)$ if M is deterministic
  - $O(|K|^2|w|)$ if M is nondeterministic

- Typical application of regular languages: pattern matching

$$L_x = \{w \in \Sigma^* : x \text{ is a substring of } w\}$$

- Regular languages can be described by regular expressions, finite automata (deterministic or not), and any combination of union, concatenation, intersection, complement, Kleene star of the above
- Languages that are not regular can be found using a pumping theorem:

Theorem (Pumping regular languages)

Let L be a regular language. Then there exists $n \geq 1$ such that any $w \in L$ with $|w| \geq n$ can be written as $w = xyz$ with

- $y \neq \varepsilon$
- $|xy| \leq n$
- $xy^iz \in L$ *for all* $i \geq 0$

Trivial proof using the pigeonhole principle

- Typical examples of non-regular languages: $\{a^nb^n : n \geq 0\}$, $\{a^p : p \text{ is prime}\}$, $\{a^nb^nc^m : n, m \geq 0\}$