CS244: Theory of Computation

Fu Song
ShanghaiTech University

Fall 2022

Part I
Automata and Languages

Outline

What ia Automata Theory

Why to Bother with Automata Theory?

Historical Perspective of Automata Theory

Chomsky Hierarchy

Regular Languages and Finite Automata
- DFA, NFA, ε-NFA, RE, RLG and Their relationship
- Non-Regular Languages
- Pumping Lemma for Regular Languages
- Myhill-Nerode Theorem
- DFA Minimization
- Closure Properties

Automata

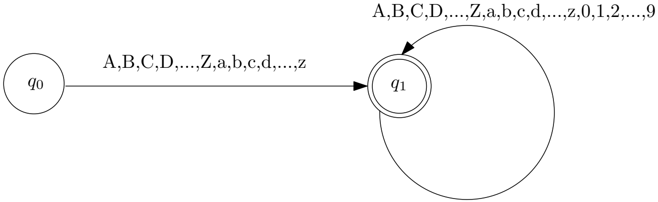
- The origin of the word "automata":

  The Greek word "$\alpha\dot{\text{u}}\tau\acute{o}\mu\alpha\tau\alpha$", which means "self-acting"

- Automata are abstract models for different aspects of computation
  - Sequential computation
    - Finite memory: Finite (state) automata
    - Finite memory + Stack: Pushdown automata
    - Unrestricted: Turing machines
  - Concurrent and reactive systems (outside of the textbook)
    - Nonterminating (ω-words): Büchi automata
    - Nondeterministic (ω-trees): Büchi tree automata
  - Rewriting systems (outside of the textbook)
    - Terms: Tree automata over ranked trees
  - Semistructured data (outside of the textbook)
    - XML documents: Tree automata over unranked trees
- Not included automata models:
  - Automata models for timed and hybrid systems
  - Automata over infinite alphabets, e.g., register automata, data automata, symbolic automata

Automata: Example

Finite automaton for identifiers of programming languages



Automata Theory

- In theoretical computer science, automata theory is the study of mathematical properties of abstract computing machines
- More specifically
  - Expressibility: Class of languages (computational problems) defined in the model

    What the model can and cannot do ?

  - Closure properties: Closed under the different operations, e.g. union and complement

    The mathematical structure of
    the class of languages defined in the model

  - Decidability and complexity: Are the decision problems (e.g. membership, nonemptiness, inclusion) decidable?
    Can they be solved in PTIME ?

    Are there (efficient) algorithms for
    the statical analysis of the model ?

Outline

What ia Automata Theory

Why to Bother with Automata Theory?

Historical Perspective of Automata Theory

Chomsky Hierarchy

Regular Languages and Finite Automata
- DFA, NFA, ε-NFA, RE, RLG and Their relationship
- Non-Regular Languages
- Pumping Lemma for Regular Languages
- Myhill-Nerode Theorem
- DFA Minimization
- Closure Properties

Why to Bother with Automata Theory?

- Theoretical foundations of various branches of computer science
  - Origin of computer science: Turing machine
  - Compiler design: Lexical analysis (Finite state automata), Syntactical analysis (Pushdown automata), Code selection (Tree automata)
  - Foundations of model checking: Büchi automata, Rabin tree automata
  - Foundations of Web data (XML document) processing: Automata over unranked trees
  - ...
- Abstract and fundamental

  Compared to programming languages,
  automata theory is more abstract,
  thus ease the mathematical reasoning,
  but still reflects the essence of computation

- Combinatorial, algorithmic, and challenging

Outline

What ia Automata Theory

Why to Bother with Automata Theory?

Historical Perspective of Automata Theory

Chomsky Hierarchy

Regular Languages and Finite Automata
- DFA, NFA, ε-NFA, RE, RLG and Their relationship
- Non-Regular Languages
- Pumping Lemma for Regular Languages
- Myhill-Nerode Theorem
- DFA Minimization
- Closure Properties

A Pioneer of Automata Theory

Alan Turing (1912-1954)

- Father of computer science
- English logician
- Propose
  Turing machine
  as
  a mathematical model
  of
  computation
- Codebreaker
  for
  German Enigma machine
  in
  World War II
- Many other pioneering work, e.g.
  Turing test

[1]

[1] On Computable Numbers, with an Application to the Entscheidungsproblem

Historical Perspective of Automata Theory

1930s	Turing machines (A. Turing)
1940s -1950s	Finite automata (W. McCulloch, W. Pitts, S. Kleene, etc.) Chomsky hierarchy (N. Chomsky)
1960s -1970s	Pushdown automata (A.G. Oettinger, M.P. Schutzenberger) Büchi automata over ω-words (J. R. Büchi) Rabin tree automata over ω-trees (M. O. Rabin) Tree automata (J. E. Doner, J. W. Thatcher, J. B. Wright, etc.)
1980s -1990s	ω-automata applied to formal verification (M. Vardi, P. Wolper, O. Kupferman, etc.)
2000s-2010s	Automata over unranked trees applied to XML (A. Bruggemann-Klein, M. Murata, D. Wood, F. Neven, etc.) Visibly pushdown automata (R. Alur, P. Madhusudan)

Remark: Only include automata models explicitly referred to in this course

Outline

What ia Automata Theory

Why to Bother with Automata Theory?

Historical Perspective of Automata Theory

Chomsky Hierarchy

Regular Languages and Finite Automata
- DFA, NFA, ε-NFA, RE, RLG and Their relationship
- Non-Regular Languages
- Pumping Lemma for Regular Languages
- Myhill-Nerode Theorem
- DFA Minimization
- Closure Properties

Chomsky Hierarchy

- A formal grammar $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$,
  - $\mathcal{N}$: a finite set of nonterminals
  - Σ: a finite set of terminals (or letters)
  - $\mathcal{P}$: a finite set of production rules
    $$\alpha \to \beta$$
    where $\alpha \in (\mathcal{N} \cup \Sigma)^+, \beta \in (\mathcal{N} \cup \Sigma)^*$
  - $S \in \mathcal{N}$: a start symbol
- Derivation relation: If $\alpha \to \beta$, then for any $w_1, w_2 \in (\mathcal{N} \cup \Sigma)^*$
  $$w_1 \alpha w_2 \models w_1 \beta w_2$$
- The language generated by G (denoted by $L(G)$):
  $\{w \in \Sigma^* \mid S \models^* w\}$.
- Grammars: $\alpha, \beta, \gamma \in (\mathcal{N} \cup \Sigma)^*$, $A, B \in \mathcal{N}$ and $a \in \Sigma$
  - Type-0 (Phrase-structure grammar): $\alpha \to \beta$ (no restrictions)
  - Type-1 (Context-sensitive grammar): $\alpha A \beta \to \alpha \gamma \beta$ such that $\gamma \neq \varepsilon$
  - Type-2 (Context-free grammar): $A \to \gamma$
  - Type-3 (Right linear grammar): $A \to a$ and $A \to aB$

  Note: $A \to \varepsilon$ is allowed in Type-1 and Type-3 only if A is S and does not occur on the right-side of any rules

Chomsky Hierarchy

Grammar	Languages	Automata	Example
Type-0	Recursively enumerable	Turing machine (TM)	$\{w \mid w$ describes a terminating TM$\}$
Type-1	Context-sensitive	Linear-bounded nondet. TM	$\{a^n b^n c^n \mid n \geq 0\}$
Type-2	Context-free	nondet. pushdown automata	$\{a^n b^n \mid n \geq 0\}$
Type-3	Regular	Finite automata	$\{a^n \mid n \geq 0\}$

Strictness of the inclusion

- Context-sensitive $\subset$ Recursive $\subset$ Recursively enumerable,
- Context-sensitive and non-context-free: $\{a^n b^n c^n \mid n \in \mathbb{N}\}$,
- Context-free and non-regular: $\{a^n b^n \mid n \in \mathbb{N}\}$.

Outline

What ia Automata Theory

Why to Bother with Automata Theory?

Historical Perspective of Automata Theory

Chomsky Hierarchy

Regular Languages and Finite Automata
- DFA, NFA, ε-NFA, RE, RLG and Their relationship
- Non-Regular Languages
- Pumping Lemma for Regular Languages
- Myhill-Nerode Theorem
- DFA Minimization
- Closure Properties

Outline

What ia Automata Theory

Why to Bother with Automata Theory?

Historical Perspective of Automata Theory

Chomsky Hierarchy

Regular Languages and Finite Automata
- DFA, NFA, ε-NFA, RE, RLG and Their relationship
- Non-Regular Languages
- Pumping Lemma for Regular Languages
- Myhill-Nerode Theorem
- DFA Minimization
- Closure Properties

Deterministic Finite Automata

Definition

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

1. Q is a finite set called the states,
2. Σ is a finite set called the alphabet,
3. $\delta : Q \times \Sigma \to Q$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

Formal Definition of Computation

Let $M = \big(Q, \Sigma, \delta, q_0, F\big)$ be a finite automaton and let $w = w_1 w_2 \cdots w_n$ be a string with $w_i \in \Sigma$ for all $i \in [n]$. Then M accepts w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for $i = 0, \ldots, n-1$, and
3. $r_n \in F$.

We say that M recognizes L if

$$L = \big\{ w \;\big|\; M \text{ accepts } w \big\}.$$

Definition

A language is called regular if some finite automaton recognizes it.

Nondeterministic Finite Automata

Definition

A nondeterministic finite automaton (NFA) is a 5-tuple $\big(Q, \Sigma, \delta, q_0, F\big)$, where

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

Let $N = \big(Q, \Sigma, \delta, q_0, F\big)$ be an NFA and let $w = y_1 y_2 \cdots y_m$ be a string with $y_i \in \Sigma$ for all $i \in [m]$. Then N accepts w if a sequence of states $r_0, r_1, \ldots, r_m$ in Q exists with:

1. $r_0 = q_0$,
2. $r_{i+1} \in \delta(r_i, y_{i+1})$ for $i = 0, \ldots, m-1$, and
3. $r_m \in F$.

Nondeterministic Finite Automata with ε-transtions

Definition

An ε-NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ is the transition function, where $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an ε-NFA and let $w = y_1 y_2 \cdots y_m$ be a string with $y_i \in \Sigma_\varepsilon$ for all $i \in [m]$. Then N accepts w if a sequence of states $r_0, r_1, \ldots, r_m$ in Q exists with:

1. $r_0 = q_0$,
2. $r_{i+1} \in \delta(r_i, y_{i+1})$ for $i = 0, \ldots, m-1$, and
3. $r_m \in F$.

Note: y_i may be ε, and $w\varepsilon = w = \varepsilon w$

Equivalence of NFAs and ε-NFAs

Theorem

$$\varepsilon\text{-}NFA \equiv NFA$$

Because an NFA is an ε-NFA, it suffices to prove that

from any ε-NFA N, an equivalent NFA N' can be constructed.

Proof

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an ε-NFA.
Compute inductively the ε-reachability relation R as follows.

- $R_0 = \{(q, q) \mid q \in Q\} \cup \{(q, q') \mid (q, \varepsilon, q') \in \delta\}$,
- $R_{i+1} = R_i \cup \{(q, q') \mid \exists q'' \in Q, (q, q''), (q'', q') \in R_i\}$.

Then let $N' = (Q, \Sigma, \delta', q_0, F')$ such that

$$\delta' = \{(q, a, q') \mid a \in \Sigma, \exists q''.(q, q'') \in R, (q'', a, q') \in \delta\}.$$

$$F' = \{q \in Q \mid \exists q' \in F.(q', q) \in R \vee (q, q') \in R\}.$$

$$a_{j+1}$$

$$q_0\ a_1\ q_1 \cdots \underbrace{q_i\ \varepsilon\ q_{i+1} \cdots q_{j-1}\ \varepsilon\ q_j}_{(q_i, q_j) \in R}\ a_{j+1}\ q_{j+1} \cdots q_n$$

Equivalence of NFAs and DFAs

Theorem

$$NFA \equiv DFA.$$

Because an DFA is an NFA, it suffices to prove that

from any NFA N, an equivalent DFA M can be constructed.

Proof

Let $N = \big(Q, \Sigma, \delta, q_0, F\big)$ be the NFA recognizing some language L. We construct a DFA $M = \big(Q', \Sigma, \delta', q_0', F'\big)$ recognizing the same L.

1. $Q' = \mathcal{P}(Q)$.
2. Let $R \in Q'$ and $a \in \Sigma$. Then we define

$$\delta'(R, a) = \big\{ q \in Q \;\big|\; q \in \delta(r, a) \text{ for some } r \in R \big\}.$$

3. $q_0' = \{q_0\}$.
4. $F' = \big\{ R \in Q' \;\big|\; R \cap F \neq \emptyset \big\}$.

Regular Expression

Syntax of regular expressions is defined by the following rules,

$$r := a \mid \varepsilon \mid \emptyset \mid r_1 \cup r_2 \mid r_1 \circ r_2 \mid r_1^*.$$

Semantics,

- $L(a) = \{a\}$, $L(\varepsilon) = \{\varepsilon\}$, $L(\emptyset) = \emptyset$,
- $L(r_1 \cup r_2) = L(r_1) \cup L(r_2)$,
- $L(r_1 \circ r_2) = L(r_1) \circ L(r_2)$,
- $L(r_1^*) = (L(r_1))^*$,

where

- $L_1 \circ L_2 = \{uv \mid u \in L_1, v \in L_2\}$ $(L \circ \emptyset = \emptyset \circ L = \emptyset)$,
- $L^* = \bigcup_{i=0}^{\infty} L^i$,
- $L^0 = \{\varepsilon\}$, $L^i = L^{i-1} \circ L$ for any $i > 0$.

We often write $r_1 r_2$ instead of $(r_1 \circ r_2)$ if no confusion arises.

Equivalence with Finite Automata

Theorem

Regular expression $\equiv$ *NFA*.

From Regular Expression to NFA

Let r be a regular expression. Construct an NFA N from r inductively as follows.

- If $r = a$, $N = (\{q_0, q_1\}, \Sigma, \delta, q_0, \{q_1\})$ such that $\delta(q_0, a) = \{q_1\}$ and $\delta(q, b) = \emptyset$ for all $q \neq q_0$ or $b \neq a$
- If $r = \varepsilon$, $N = (\{q_0\}, \Sigma, \delta, q_0, \{q_0\})$ such that $\delta(q_0, a) = \emptyset$ for all $a \in \Sigma$
- If $r = \emptyset$, $N = (\{q_0\}, \Sigma, \delta, q_0, \emptyset)$ such that $\delta(q_0, a) = \emptyset$ for all $a \in \Sigma$
- if $r = r_1 \cup r_2$, let $N_i = (Q_i, \Sigma, \delta_i, q_0^i, F_i)$ for r_i (where $i = 1, 2$), then $N = (Q_1 \cup Q_2 \cup \{q_0\}, \Sigma, \delta, q_0, F)$ such that
  - $\delta = \delta_1 \cup \delta_2 \cup \{(q_0, a, q) \mid (q_0^1, a, q) \in \delta_1$ or $(q_0^2, a, q) \in \delta_2\}$,
  - $F = F_1 \cup F_2$.
- If $r = r_1 \circ r_2$, let $N_i = (Q_i, \Sigma, \delta_i, q_0^i, F_i)$ for r_i (where $i = 1, 2$), then $N = (Q_1 \cup Q_2, \Sigma, \delta, q_0^1, F)$ such that
  - $\delta = \delta_1 \cup \delta_2 \cup \{(q, a, q') \mid q \in F_1, (q_0^2, a, q') \in \delta_2\}$,
  - if $q_0^2 \in F_2$, then $F = F_2 \cup F_1$, otherwise $F = F_2$.
- If $r := (r_1)^*$, let $N_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$ for r_1, then $N = (Q_1 \cup \{q_0\}, \Sigma, \delta, q_0, \{q_0\})$ such that
  $\delta = \delta_1 \cup \{(q_0, a, q) \mid (q_0^1, a, q) \in \delta_1\} \cup \{(q, a, q_0) \mid \exists q' \in F_1.(q, a, q') \in \delta_1\}$.

Note: $Q_1 \cap Q_2 = \emptyset$

From NFA to Regular Expression

We need generalized nondeterministic finite automata (GNFA) – nondeterministic finite automata wherein the transition arrows may have any regular expressions as labels.

1. The start state has transition arrows going to every other state but no arrows coming in from any other state.

2. There is only a single accept state, and it has arrows coming in from every other state but no arrows going to any other state. Furthermore, the accept state is not the same as the start state.

3. Except for the start and accept states, one arrow goes from every state to every other state and also from each state to itself.

Generalized Nondeterministic Finite Automata

Definition

A GNFA is a 5-tuple $(Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$, where

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta : \big(Q - \{q_{\text{accept}}\}\big) \times \big(Q - \{q_{\text{start}}\}\big) \to \mathcal{R}$ is the transition function, where $\mathcal{R}$ is the set of regular expressions,
4. q_{start} is the start state, and
5. q_{accept} is the accept state.

A GNFA accepts a string $w \in \Sigma^*$ if $w = w_1 w_2 \ldots w_k$, where each $w_i \in \Sigma^*$ and a sequence of states $q_0, q_1, \ldots, q_k$ exists such that

1. $q_0 = q_{\mathrm{start}}$ is the start state,
2. $q_k = q_{\mathrm{accept}}$ is the accept state, and
3. for each $i \in [k]$, we have $w_i \in L(R_i)$, where $R_i = \delta(q_{i-1}, q_i)$.

From DFA to Regular Expression

Let M be the DFA for language L.

- We convert M to a GNFA G by adding a new start state and a new accept state and additional transition arrows as necessary.
- Then we use a procedure CONVERT on G to return an equivalent regular expression.

CONVERT(G):

1. Let k be the number of states of G.
2. If $k = 2$, then return the regular expression R labelling the arrow from q_{start} to q_{accept}.
3. If $k > 2$, we select any state $q_{\text{rip}} \in Q - \{q_{\text{start}}, q_{\text{accept}}\}$ and let $G' = (Q', \Sigma, \delta', q_{\text{start}}, q_{\text{accept}})$ be the GNFA, where

$$Q' = Q - \{q_{\text{rip}}\},$$

and for any $q_i \in Q' - \{q_{\text{accept}}\}$ and $q_j \in Q' - \{q_{\text{start}}\}$, let

$$\delta'(q_i, q_j) = ((r_1)(r_2)^*(r_3)) \cup (r_4),$$

for $r_1 = \delta(q_i, q_{\text{rip}})$, $r_2 = \delta(q_{\text{rip}}, q_{\text{rip}})$, $r_3 = \delta(q_{\text{rip}}, q_j)$, and $r_4 = \delta(q_i, q_j)$.

4. Compute CONVERT(G') and return this value.

Right Linear Grammar

A right linear grammar is a grammar $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$ such that $\mathcal{P}$ contains rules of the form

$$A \to aB, \text{ or } A \to a, \text{ or } A \to \varepsilon, \text{ where } a \in \Sigma, A, B \in \mathcal{N}.$$

Theorem

Right linear grammar $\equiv$ *NFA*

From Right Linear Grammar to NFA

Let $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$ be a right linear grammar.
Construct $N = (\mathcal{N} \cup \{q_f\}, \Sigma, \delta, S, F)$ as follows.

- $F = \{q_f\} \cup \{A \mid A \to \varepsilon\}$.
- δ is defined by the following rules,

$$(A, a, B) \in \delta \textit{ iff } A \to aB, \quad (A, a, q_f) \in \delta \textit{ iff } A \to a.$$

From NFA to Right Linear Grammar

Let $N = (Q, \Sigma, \delta, q_0, F)$ be a NFA.
Construct a right linear grammar $G = (Q, \Sigma, \mathcal{P}, q_0)$ as follows.

$$q \to aq' \textit{ iff } (q, a, q') \in \delta, \quad q \to \varepsilon \textit{ iff } q \in F.$$

Corollary

Right linear grammar$\equiv$*Regular expression*$\equiv$$\varepsilon$*-NFA*$\equiv$*NFA*$\equiv$*DFA.*

Outline

What ia Automata Theory

Why to Bother with Automata Theory?

Historical Perspective of Automata Theory

Chomsky Hierarchy

Regular Languages and Finite Automata
- DFA, NFA, ε-NFA, RE, RLG and Their relationship
- Non-Regular Languages
- Pumping Lemma for Regular Languages
- Myhill-Nerode Theorem
- DFA Minimization
- Closure Properties

Languages Need Counting

Suppose $\Sigma = \{0, 1\}$.

- $C = \left\{ w \in \{0,1\}^* \;\middle|\; w \text{ has an equal number of 0s and 1s} \right\}$.
- $D = \left\{ w \in \{0,1\}^* \;\middle|\; w \text{ has an equal number of occurrences of 01 and 10 as substrings} \right\}$,

  e.g., $010 \in D$.

Which one is regular ?

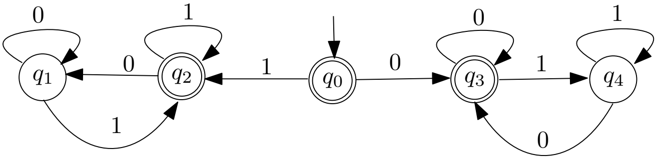
Lemma

C is not regular and D is regular.

- How do we PROVE that a language is regular?
- How do we PROVE that a language is NOT regular?

PROVE that a Language is (NOT) Regular

- How do we PROVE that a language is regular?
  - Construct an automaton for that language
  - E.g., the automaton for D



- How do we PROVE that a language is NOT regular?
  - It is impossible to construct an automaton for that language
  - Basic technique: prove by contradiction: assume language is regular and get a contradiction
  - We will prove all Regular Languages have the pumping property
  - To show that a language is not regular, we show that the language does NOT have the pumping property

Outline

What ia Automata Theory

Why to Bother with Automata Theory?

Historical Perspective of Automata Theory

Chomsky Hierarchy

Regular Languages and Finite Automata
- DFA, NFA, ε-NFA, RE, RLG and Their relationship
- Non-Regular Languages
- Pumping Lemma for Regular Languages
- Myhill-Nerode Theorem
- DFA Minimization
- Closure Properties

The Pumping Lemma for Regular Languages

Lemma

If L is a regular language, then there is an integer $p \geq 1$ (i.e., the pumping length) where if s is any string in A of length at least p (i.e., $|s| \geq p$), then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. *$\forall i \geq 0$, $xy^i z \in L$ (pumping the loop y produces strings that are in the language),*
2. *$|y| > 0$ (the pumped section is not empty), and*
3. *$|xy| \leq p$ (a loop must appear within the first p states).*

Any string xyz in L can be pumped along y.

Using the Pumping Lemma to show a language is Non-Regular

- Assume language L is regular (then find a contradiction)
- If L is regular, then the pumping conditions hold for EVERY string in L that is long enough
- Show that the language is NOT regular by finding SOME STRING, call it s (choosing s is VERY important and constructive)
- We must assume that s satisfies any two of pumping conditions, show that the other one does not holds
  - We usually assume that conditions 2 and perhaps 3 hold
  - then show it is IMPOSSIBLE to divide s into pieces (i.e. $s = xyz$) that meet the pumping conditions (usually condition 1)
  - If you can find a string that is impossible to divide this way, then the language cannot be regular!
- Remember, you only need to find one string that cannot be pumped!

Proof

Let $M = \big(Q, \Sigma, \delta, q_1, F\big)$ be a DFA recognizing L and $p := |Q|$.

Let $s = s_1 s_2 \cdots s_n$ be a string in L with $n \geq p$. Let $r_1, \ldots, r_{n+1}$ be the sequence of states that M enters while processing s, i.e.,

$$r_{i+1} = \delta(r_i, s_i), \forall i \in [n].$$

Among the first $p+1$ states in the sequence, two must be the same, say r_j and r_ℓ with $j < \ell \leq p+1$. We define

$$x = s_1 \cdots s_{j-1}, y = s_j \cdots s_{\ell-1}, \text{ and } z = s_\ell \cdots s_n.$$

Then,

1. $\forall i \geq 0$, $xy^i z = s_1 \cdots s_{j-1}(s_j \cdots s_{\ell-1})^i s_\ell \cdots s_n \in L$, as $r_1 \cdots r_j (r_{j+1} \cdots r_\ell)^i r_{\ell+1} \cdots r_{n+1}$ is still an accepting run of M,
2. $|y| = |s_j \cdots s_{\ell-1}| > 0$, as $j < \ell$, and
3. $|xy| = |s_1 \cdots s_{j-1} s_j \cdots s_{\ell-1}| \leq p$, as $\ell \leq p+1$.

Example (I)

Example

The language $L = \left\{ 0^n 1^n \;\middle|\; n \geq 0 \right\}$ is not regular.

Proof.

- Assume that the language L is regular.
- Choose p be the pumping length and consider $s = 0^p 1^p$. By the Pumping Lemma, $s = xyz$ with $xy^i z \in L$ for all $i \geq 0$ and $|xy| \leq p$. Then $y \in 00^*$.

  1. If $y \in 00^*$, then $xyyz$ has more 0s than 1s, a contradiction.

$\square$

Example (II)

Example

The language $C = \{ w \mid w \text{ has an equal number of 0s and 1s} \}$ is not regular.

Proof.

- Assume that the language C is regular.
- Choose p be the pumping length and consider $s = 0^p 1^p$. By the Pumping Lemma, $s = xyz$ with $|xy| \leq p$ and $xy^i z \in C$ for all $i \geq 0$. Thus $xy \in 00^*$ and the contradiction follows easily.

$\square$

Quiz

Prove that the language $L = \left\{ ww \;\middle|\; w \in \{0,1\}^* \right\}$ is not regular.

Discussion

Is the language $L = \left\{ ww^R v \;\middle|\; v, w \in \{0,1\}^+ \right\}$ regular?

- This language is not regular!
- But, we cannot prove by applying pumping lemma,
- Pumping lemma is only a necessary condition, but not a sufficient condition!
- We need Myhill-Nerode Theorem:

Language L is regular

iff

the number of equivalence classes of L is finite.

Outline

What ia Automata Theory

Why to Bother with Automata Theory?

Historical Perspective of Automata Theory

Chomsky Hierarchy

Regular Languages and Finite Automata
- DFA, NFA, ε-NFA, RE, RLG and Their relationship
- Non-Regular Languages
- Pumping Lemma for Regular Languages
- Myhill-Nerode Theorem
- DFA Minimization
- Closure Properties

Myhill-Nerode Equivalence Relation

Let $L \subseteq \Sigma^*$. Define $\sim_L$ over Σ^* as follows: For any $u, v \in \Sigma^*$,

$$u \sim_L v \text{ iff } \forall z \in \Sigma^*. uz \in L \Leftrightarrow vz \in L.$$

Proposition. $\sim_L$ is a right congruence.
Note: A right congruence on a group is an equivalence relation on the group with the property that the equivalence relation is preserved on right multiplication by any element of the group.

Proof.

- $\sim_L$ is an equivalence relation: reflexive, transitive and symmetric.
- For any $u, v : u \sim_L v$ and $z \in \Sigma^*$, $uz \sim_L vz$.

□

The index of $\sim_L$ is the number of equivalence classes of $\sim_L$.

Example: Let $L = a^*b$, then $\sim_L$ contains

three equivalence classes a^*, a^*b, $a^*b(a \cup b)^+$.

Myhill-Nerode Theorem

Theorem. Let $L \subseteq \Sigma^*$. Then L is regular iff $\sim_L$ is of finite index.

- ($\Rightarrow$) Suppose L is defined by the NFA $N = (Q, \Sigma, \delta, q_0, F)$.
  Then for any $x \in \Sigma^*$, define $R(x)$ as
  the set of states that can be reached from q_0 after reading x.

  It follows that
  for any $u, v \in \Sigma^*$, $R(u) = R(v) \Rightarrow u \sim_L v$.

  The index of $\sim_L$ is at most as many as $|\{R(v) \mid v \in \Sigma^*\}| \leq 2^{|Q|}$.
  Therefore, $\sim_L$ is of finite index.

- ($\Leftarrow$) Suppose $\sim_L$ is of finite index.
  Let $E_1, \ldots, E_n$ be the equivalence classes of $\sim_L$ and $E_1 = [\varepsilon]$.
  Then $N = (\{E_1, \ldots, E_n\}, \Sigma, \delta, E_1, \{E_i \mid E_i \cap L \neq \emptyset\})$, where
  $(E_i, a, E_j) \in \delta$ *iff* $\exists u \in E_i . ua \in E_j$.

Example

Prove that the language $L = \left\{0^n 1^n \mid n \geq 0\right\}$ is not regular using the Myhill-Nerode Theorem

We showed that the index of $\sim_L$ is not finite.

Consider the sequence of strings $x_1, x_2 \cdots$ such that $x_i = 0^i$ for $i \geq 1$.

We now see that no two of these are equivalent to each other with respect to $\sim_L$. Consider $x = 0^i$ and $y = 0^j$ such that $i \neq j$, and let $z = 1^i$. Then, $xz \in L$ but $yz \notin L$, hence $x \not\sim_L y$.

Taking Home Quiz

Prove that the language $L = \left\{ ww^R v \;\middle|\; v, w \in \{0,1\}^+ \right\}$ is not regular using the Myhill-Nerode Theorem

Hint: find an infinite sequence of strings $x_1, x_2, \cdots$ and prove that they are not equivalent to each other with respect to $\sim_L$.

Outline

What ia Automata Theory

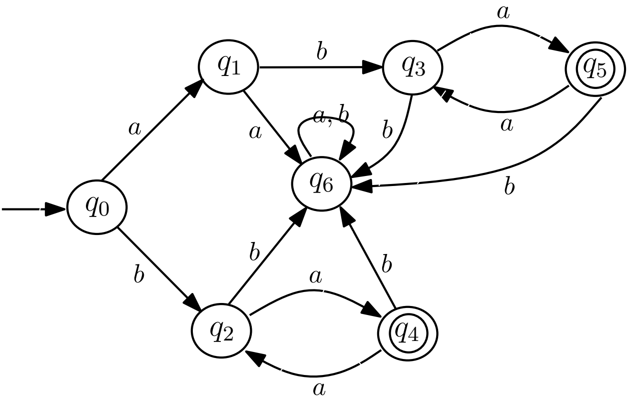
Why to Bother with Automata Theory?

Historical Perspective of Automata Theory

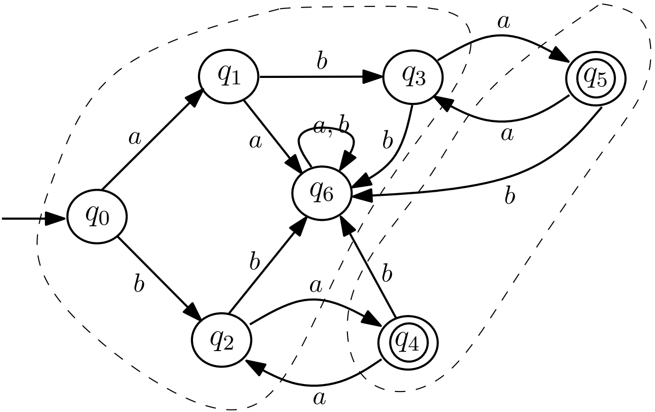
Chomsky Hierarchy

Regular Languages and Finite Automata
- DFA, NFA, ε-NFA, RE, RLG and Their relationship
- Non-Regular Languages
- Pumping Lemma for Regular Languages
- Myhill-Nerode Theorem
- DFA Minimization
- Closure Properties

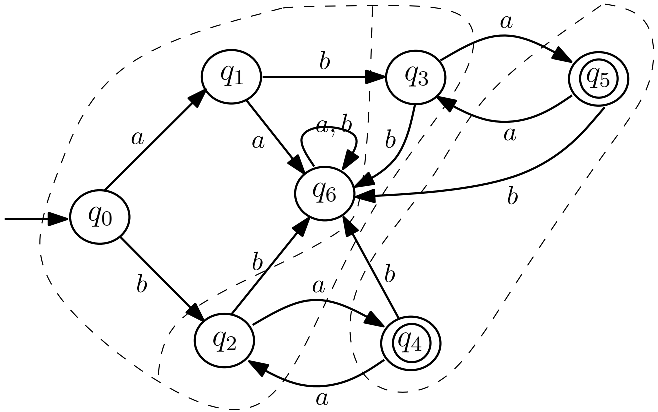
Minimization: An example



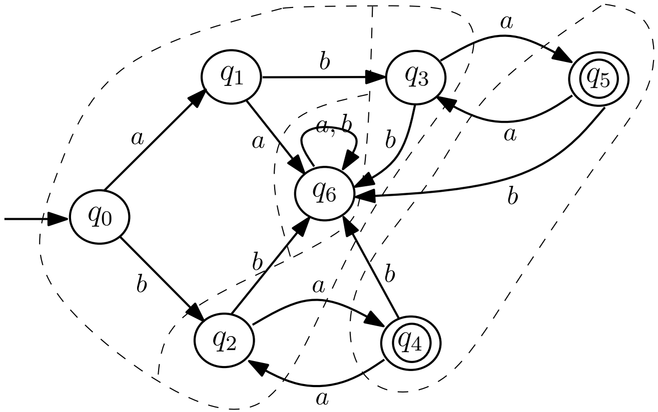
Minimization: An example



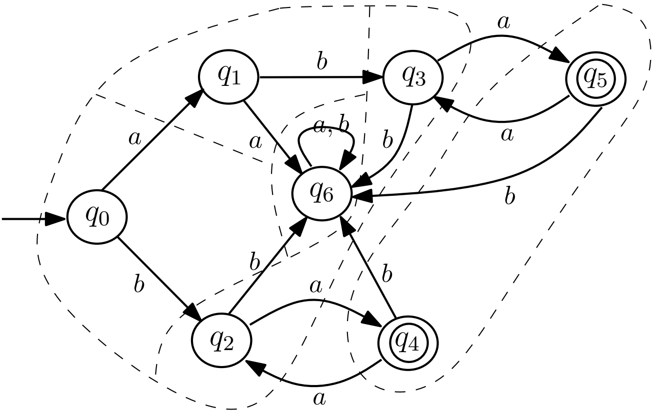
Minimization: An example



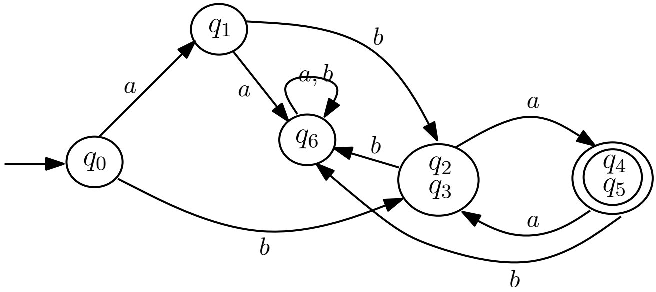
Minimization: An example



Minimization: An example



Minimization: An example



Uniqueness of minimum-size DFA

Theorem. For every regular language $L \subseteq \Sigma^*$,

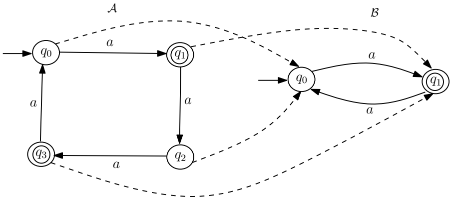
there is a unique complete DFA of the minimum size.

Morphism

Let $M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$ be two complete DFAs. A morphism from M_1 to M_2 is a (surjective) mapping h from Q_1 to Q_2 such that

- $h(q_0^1) = q_0^2$,
- for every $q \in Q_1$, $q \in F_1$ iff $h(q) \in F_2$,
- for every $q, q' \in Q_1, a \in \Sigma$ s.t. $\delta_1(q, a) = q'$, it holds $\delta_2(h(q), a) = h(q')$.

A morphism is called an isomorphism iff h is bijective.



Uniqueness of minimum-size DFA: continued

Let $L \subseteq \Sigma^*$ be a regular language.
Let $M_L = (Q_L, \Sigma, \delta_L, q_0^L, F_L)$ be the DFA corresponding to $\sim_L$,

- Q_L is the set of equivalence classes of $\sim_L$,
- $\delta_L([x], a) = [xa]$ for any $x \in \Sigma^*$,
- $q_0^L = [\varepsilon]$, $F_L = \{[x] \mid x \in L\}$.

Claim. $\forall$ complete DFA M such that $L(M) = L$, $\exists$ a morphism from M to M_L.

Proof (1)

Claim. $\forall$ complete DFA M such that $L(M) = L$, $\exists$ a morphism from M to M_L.

Proof.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a complete DFA such that $L(M) = L$.
Then for every $x, y \in \Sigma^*$ such that $\delta(q_0, x) = \delta(q_0, y)$, we have

for every $z \in \Sigma^*$, $xz \in L$ iff $yz \in L$, so $x \sim_L y$.

Define a mapping $h : Q \to Q_L$ as follows: $h(q) = [x]$ s.t. $\delta(q_0, x) = q$.

- h is surjective since M is complete,
- $h(q_0) = [\varepsilon] = q_0^L$,
- if $q \in F$, then $h(q) = [x]$ for $x \in \Sigma^*$ s.t. $\delta(q_0, x) = q$. So $x \in L$, $[x] \in F_L$,
- if $\delta(q, a) = q'$, then $\delta_L(h(q), a) = \delta_L([x], a) = [xa] = h(q')$ with $\delta(q_0, x) = q$.

$\square$

Proof (2)

Because $|M| \geq |M_L|$,

it follows that M_L is the DFA defining L of the minimum size.

Uniqueness.

Suppose M' is a DFA of the minimum size defining L.

Then M' has the same size as M_L.

According to the claim, there is a morphism h from M' to M_L.

Because h is surjective, it follows that h is bijective.

Therefore, h is an isomorphism from M' to M_L.

Minimization

Given a DFA M, construct an equivalent DFA M' of the minimum size.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Compute inductively an equivalence relation $\approx_M$ over Q as follows, until $\approx_M^i = \approx_M^{i+1}$.

- $q \approx_M^0 q'$ iff $q \in F \Leftrightarrow q' \in F$,
- $q \approx_M^{i+1} q'$ iff $q \approx_M^i q'$ and $\forall a \in \Sigma$, $\delta(q, a) \approx_M^i \delta(q', a)$.

Because $\forall i.\ \approx_M^{i+1} \subseteq \approx_M^i$, it follows that the above procedure terminates.

Observation. $\approx_M$ enjoys the following properties.

- $q \approx_M q' \Rightarrow q \in F$ iff $q' \in F$ and $\forall x \in \Sigma^*$, $\delta(q, x) \approx_M \delta(q', x)$,
- $q \not\approx_M q' \Rightarrow \exists x \in \Sigma^*$ s.t. $\delta(q, x) \in F$ and $\delta(q', x) \notin F$ or vice versa.

Corollary. $\forall u, v \in \Sigma^*$, $u \sim_L v$ iff $\delta(q_0, u) \approx_M \delta(q_0, v)$.
Therefore,

$M/\approx_M$ is the DFA of the minimum size.

Outline

What ia Automata Theory

Why to Bother with Automata Theory?

Historical Perspective of Automata Theory

Chomsky Hierarchy

Regular Languages and Finite Automata
- DFA, NFA, ε-NFA, RE, RLG and Their relationship
- Non-Regular Languages
- Pumping Lemma for Regular Languages
- Myhill-Nerode Theorem
- DFA Minimization
- Closure Properties

The Regular Operators

Definition

Let L_1 and L_2 be languages. We define the regular operations union, concatenation, and star as follows:

- Union: $L_1 \cup L_2 = \{x \mid x \in L_1 \text{ or } x \in L_2\}$.
- Intersection: $L_1 \cap L_2 = \{x \mid x \in L_1 \text{ and } x \in L_2\}$.
- Concatenation: $L_1 \circ L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$.
- Star: $L^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in L\}$.
- Complementation: $\overline{L} = \Sigma^* - L$.

Closure under Union

Theorem

The class of regular languages is closed under the union operation.

In other words, if L_1 and L_2 are regular languages, so is $L_1 \cup L_2$.

Closure under Union

For $i \in [2]$ let $N_i = \left(Q_i, \Sigma_i, \delta_i, q_i, F_i\right)$ recognize L_i. We construct an ε-NFA $N = \left(Q, \Sigma, \delta, q_0, F\right)$ to recognize $L_1 \cup L_2$:

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$.
2. q_0 is the start state.
3. $F = F_1 \cup F_2$.
4. For any $q \in Q$ and any $a \in \Sigma_\varepsilon$

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \\ \delta_2(q, a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$

Second Proof of the Closure under Union

For $i \in [2]$ let $M_i = \big(Q_i, \Sigma_i, \delta_i, q_i, F_i\big)$ recognize L_i. We can assume without loss of generality $\Sigma_1 = \Sigma_2$:

- Let $a \in \Sigma_2 - \Sigma_1$.
- We add $\delta_1(r, a) = r_{\mathrm{trap}}$, where r_{trap} is a new state with

$$\delta_1(r_{\mathrm{trap}}, w) = r_{\mathrm{trap}}$$

  for every w.

Proof (2)

We construct $M = \big(Q, \Sigma, \delta, q_0, F\big)$ to recognize $L_1 \cup L_2$:

1. $Q = Q_1 \times Q_2 = \big\{(r_1, r_2) \bigm| r_1 \in Q_1 \text{ and } r_2 \in Q_2\big\}$.
2. $\Sigma = \Sigma_1 = \Sigma_2$.
3. For each $(r_1, r_2) \in Q$ and $a \in \Sigma$ we let

$$\delta\big((r_1, r_2), a\big) = \big(\delta_1(r_1, a), \delta_2(r_2, a)\big).$$

4. $q_0 = (q_1, q_2)$.
5. $F = (F_1 \times Q_2) \cup (Q_1 \times F_2) = \big\{(r_1, r_2) \bigm| r_1 \in F_1 \text{ or } r_2 \in F_2\big\}$.

Closure Under Intersection

Theorem

The class of regular languages is closed under the intersection operation.

Proof

We construct $M = \big(Q, \Sigma, \delta, q_0, F\big)$ to recognize $L_1 \cap L_2$:

1. $Q = Q_1 \times Q_2 = \big\{(r_1, r_2) \bigm| r_1 \in Q_1 \text{ and } r_2 \in Q_2\big\}$.
2. $\Sigma = \Sigma_1 = \Sigma_2$.
3. For each $(r_1, r_2) \in Q$ and $a \in \Sigma$ we let
$$\delta\big((r_1, r_2), a\big) = \big(\delta_1(r_1, a), \delta_2(r_2, a)\big).$$
4. $q_0 = (q_1, q_2)$.
5. $F = F_1 \times F_2 = \big\{(r_1, r_2) \bigm| r_1 \in F_1 \text{ and } r_2 \in F_2\big\}$.

Closure Under Concatenation

Theorem

The class of regular languages is closed under the concatenation operation.

Proof

For $i \in [2]$ let $N_i = \big(Q_i, \Sigma_i, \delta_i, q_i, F_i\big)$ recognize L_i. We construct an $N = \big(Q, \Sigma, \delta, q_1, F_2\big)$ to recognize $L_1 \circ L_2$:

1. $Q = Q_1 \cup Q_2$.
2. The start state q_1 is the same as the start state of N_1.
3. The accept states F_2 are the same as the accept states of N_2.
4. For any $q \in Q$ and any $a \in \Sigma_\varepsilon$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 - F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q,a) & q \in Q_2. \end{cases}$$

Closure Under Kleene Star

Theorem

The class of regular languages is closed under the star operation.

Proof

Let $N_1 = \left(Q_1, \Sigma, \delta_1, q_1, F_1\right)$ recognize L_i. We construct an $N = \left(Q, \Sigma, \delta, q_0, F\right)$ to recognize L_1^*:

1. $Q = \{q_0\} \cup Q_1$.
2. The start state q_0 is the new start state.
3. $F = \{q_0\} \cup F_1$.
4. For any $q \in Q$ and any $a \in \Sigma_\varepsilon$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 - F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \varepsilon \\ \{q_1\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$

Closure Under Complementation

Theorem

The class of regular languages is closed under complementation.

Proof

Let $N = \big(Q, \Sigma, \delta, q_0, F\big)$ be an DFA to recognize L.

Suppose N is complete, i.e., for every $q \in Q, a \in \Sigma$, $\delta(q, a) \neq \emptyset$.

The DFA $\overline{N} = \big(Q, \Sigma, \delta, q_0, Q - F\big)$ recognizes $\overline{L}$.