CSE 105 Review

Kevin Jacob

Winter 2025

Table Of Contents

1 Regular Expressions

1.1 Definition

Basis steps of recursive definition

- a is a regular expression, for $a \in \Sigma$
- ϵ is a regular expression
- $\emptyset$ is a regular expression

Recursive steps of recursive definition

- $(R_1 \cup R_2)$ is a regular expression when R_1, R_2 are regular expressions
  $L((R_1 \cup R_2)) = L(R_1) \cup L(R_2) = \{w | w \in L(R_1) \vee w \in L(R_2)\}$
- $(R_1 \circ R_2)$ is a regular expression when R_1, R_2 are regular expressions
  $L((R_1 \circ R_2)) = L(R_1) \circ L(R_2) = \{uv | u \in L(R_1) \wedge v \in L(R_2)\}$
- R_1^* is a regular expression when R_1 is a regular expression
  $L((R_2)^*) = (L(R_1))^* = \{w_1, \ldots, w_k | k \geq 0 \text{ and each } w_i \in L(R_1)\}$

1.2 Conventions

Assuming Σ is the alphabet, we use the following conventions

- Σ
  regular expression describing language consisting of all strings of length 1 over Σ
- $*$ then $\circ$ then $\cup$
  precedence order, unless parentheses are used to change it
- $R_1 R_2$
  shorthand for $R_1 \circ R_2$ (concatenation symbol is implicit)
- R^+
  shorthand for $R^* \circ R$
- R^k
  shorthand for R concatenated with itself k times, where k is a (specific) natural number

2 Finite Automata

2.1 Definition

The **formal definition** of a finite automata consists of a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ and a finite automata can also be represented by a state diagram

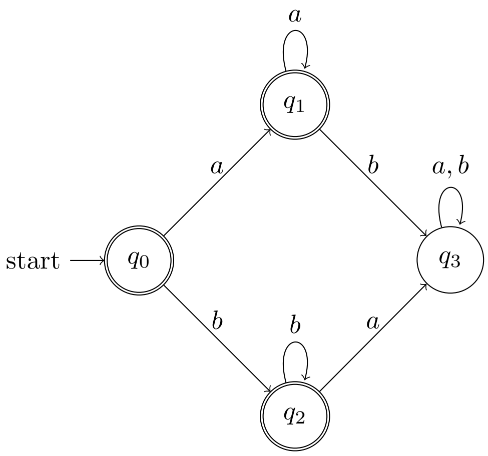
- **Finite set of states** Q can be labeled by any collection of distinct names
- The **alphabet** Σ determines the possible inputs to the automaton. Each input to the automaton is a string over Σ
- The **transition function** δ gives the next state of the automaton based on the current state of the machine and on the next input symbol
- The **start state** q_0 is an element of Q. each computation of the machine starts at the start state.
- The **accept (final) states** F form a subset of the states of the automaton, $F \subseteq Q$. These states flag if a machine accepts or rejects an input string.

The computation of a machine on an input string is a sequence of states in the machine, starting with the start state, determined by transitions of the machine as it reads successive input symbols.

The finite automaton M accepts the given input string exactly when the computation of M on the input string ends in an accept state.

The language of M, $L(M)$, is defined as the set of all strings that are each accepted by the machine M. Each string that is rejected by M is not in $L(M)$.

2.2 Example



Formal Definition: $(\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, \{q_0, q_1, Q_2\})$
where $\delta : Q \times \Sigma \to Q$

Q/Σ	a	b
q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_3	q_2
q_3	q_3	q_3

Language recognized by automaton: $L(a^* \cup b^*)$

3 Nondeterministic Automata

3.1 Definition

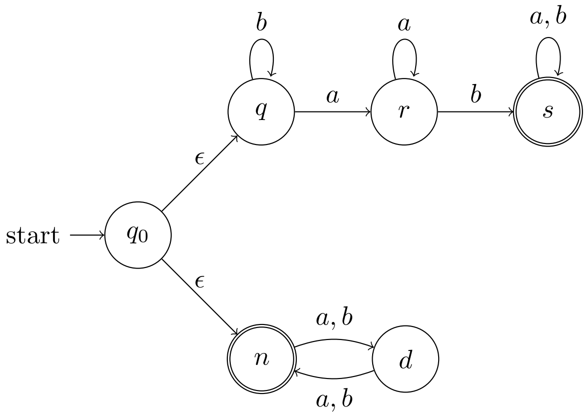
The computation of a **determinstic** finite automaton has exactly one choice for its next step given the current state and character read, but an NFA allows us to have multiple

The **formal definition** of a nondeterministic finite automaton also consists of a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ that can also be represented by a state diagram

- **Finite set of states** Q can be labelled by any collection of distinct names
- The **alphabet** Σ where each input to the automaton is a string over Σ
- The **transition function** $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ gives the **set of possible next states** for a transition from the current state upon reading a symbol or spontaneously moving
- The **start state** q_0 is an element of Q. each computation of the machine starts at the start state.
- The **accept (final) states** F form a subset of the states of the automaton, $F \subseteq Q$. These states flag if a machine accepts or rejects an input string.

M accepts the input string $w \in \Sigma^*$ if and only if **there** is a computation of M on w that processes the **whole string** and ends in an accept state

3.2 Example



Formal Definition: $(\{q_0, q, r, s, n, d\}, \{a, b\}, \delta, q_0, \{s, n\})$
where $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$

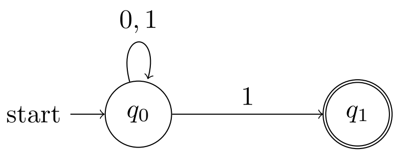
states/labels	a	b	ϵ
q_0	$\emptyset$	$\emptyset$	$\{q, n\}$
q	$\{r\}$	$\{q\}$	$\emptyset$
r	$\{r\}$	$\{s\}$	$\emptyset$
s	$\{s\}$	$\{s\}$	$\emptyset$
n	$\{d\}$	$\{d\}$	$\emptyset$
d	$\{n\}$	$\{n\}$	$\emptyset$

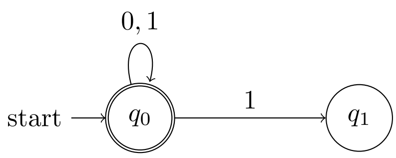
Language recongized by NFA: $\{w \in \{a, b\}^* | w$ has even length or has ab as a substring$\}$

4 Automata Constructions

4.1 Complementation

- The collection of languages that are each recognizable by a DFA is **closed** under complementation. We can flip the accept status of states in a DFA.
- The complementation construction doesn't work with an NFA





The string 1 *is accepted by both NFAs*

4.2 Union

- The collection of languages that are each recognizable by a NFA is **closed** under union. We can add a start state and add transitions of the empty string to the start of different NFAs.
- We can't do the same construction since spontaneous moves are not possible in a DFA

4.3 DFA Constructions for Union and Intersection

4.3.1 Union

Suppose A_1, A_2 are languages over the alphabet Σ. If there is a DFA M_1 such that $L(M_1) = A_1$ and a DFA M_2 such that $L(M_2) = A_2$, then there is a DFA M such that $L(M) = A_1 \cup A_2$.
Formal Construction
$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ where $L(M_1) = A_1$ and $L(M_2) = A_2$. We want to build M with $L(M) = A_1 \cup A_2$.
$M = (Q, \Sigma, \delta, q_0, F)$
$Q = \{(q, q') | q \in Q_1 \text{ and } q' \in Q_2\} = Q_1 \times Q_2$
$\delta : Q \times \Sigma \to Q \qquad \delta(((q, q'), x)) = (\delta_1(q, x), \delta_2(q', x))$
$q_0 = (q_1, q_2)$
$F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$

4.3.2 Intersection

Suppose A_1, A_2 are languages over an alphabet Σ. If there is a DFA M_1 such that $L(M_1) = A_1$ and a DFA M_2 such that $L(M_2) = A_2$, then there is a DFA M such that $L(M) = A_1 \cap A_2$
Formal Construction:
$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ where $L(M_1) = A_1$ and $L(M_2) = A_2$. We want to build M with $L(M) = A_1 \cap A_2$.
$M = (Q, \Sigma, \delta, q_0, F)$
$Q = \{(q, q') | q \in Q_1 \text{ and } q' \in Q_2\} = Q_1 \times Q_2$
$\delta : Q \times \Sigma \to Q \qquad \delta(((q, q'), x)) = (\delta_1(q, x), \delta_2(q', x))$
$q_0 = (q_1, q_2)$
$F = F_1 \times F_2$

5 Regular Languages

If a language is regular

- there is a regular expression that describes it
- there is a DFA that recognizes it
- there is a NFA that recognizes it

5.1 Concactenation

Suppose A_1, A_2 are languages over an alphabet Σ, If there is a NFA N_1 such that $L(N_1) = A_1$ and NFA N_2 such that $L(N_2) = A_2$, then there is another NFA N such that $L(N) = A_1 \circ A_2$
Formal Construction
$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ where $L(N_1) = A_1$ and $L(N_2) = A_2$. We want to build N with $L(N) = A_1 \circ A_2$.
$N = (Q, \Sigma, \delta, q_0, F)$
$Q = Q_1 \cup Q_2$
$q_0 = q_1$
$F = F_2$
$\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$

$$\delta((q, a)) = \begin{cases} \delta_1((q, a)) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1((q, a)) & q \in F_1 \text{ and } a \in \Sigma \\ \delta_1((q, a)) \cup \{q_2\} & q \in F_1 \text{ and } a = \epsilon \\ \delta_2((q, a)) & q \in Q_2 \end{cases}$$

5.2 Kleene Star

Suppose A is a language over alphabet Σ. If there is a NFA N such that $L(N) = A$, then there is another NFA N' such that $L(N') = A^*$.

Formal Construction

$N = (Q, \Sigma, \delta, q_1, F)$ where $q_0 \notin Q$. We want to build N' with $L(N') = A^*$.
$N' = (Q', \Sigma, \delta', q_0, F')$
$Q' = Q \cup \{q_0\}$
$F' = F \cup \{q_0\}$
$\delta' : Q' \times \Sigma_\epsilon \to \mathcal{P}(Q')$

$$\delta'((q,a)) = \begin{cases} \delta((q,a)) & q \in Q \text{ and } q \notin F \\ \delta((q,a)) & q \in F \text{ and } a \in \Sigma \\ \delta((q,a)) \cup \{q_1\} & q \in F \text{ and } a = \epsilon \\ \{q_1\} & q = q_0 \text{ and } a = \epsilon \\ \emptyset & q = q_0 \text{ and } a \in \Sigma \end{cases}$$

5.3 NFA to DFA

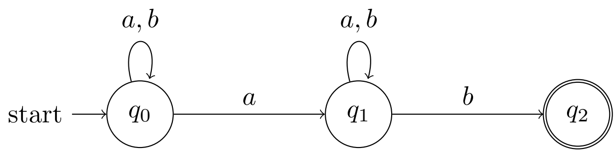
Suppose A is a language over alphabet Σ. If there is a NFA N such that $L(N) = A$ then there is a DFA M such that $L(M) = A$

Formal Construction

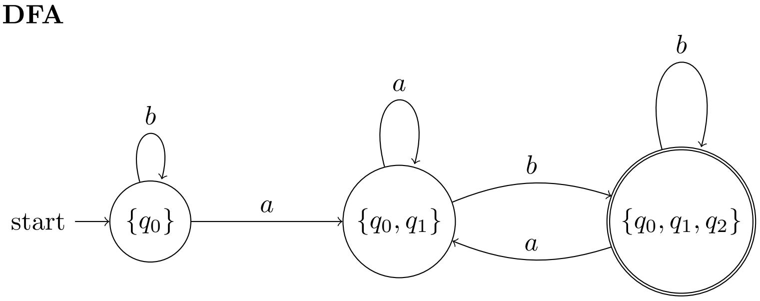
We can treat states in M as macro-states which are collections of states from N that represent the set of possible states a computation of N might be in.

Let $N = (Q, \Sigma, \delta, q_0, F)$. We want to build M with $L(M) = A$.
$M = (\mathcal{P}(Q), \Sigma, \delta', q', \{X \subseteq Q | X \cap F \neq \emptyset\})$
$q' = \{q \in Q | q = q_0$ or is accessible from q_0 by spontaneous moves in $N\}$
$\delta'((X, x)) = \{q \in \delta((r, x))$ for some $r \in X$ or is accessible from such an r by spontaneous moves in $N\}$

NFA



DFA



5.4 Regular Expression to NFA/DFA

5.4.1 Regular Expression to NFA

Suppose A is a language over an alphabet Σ. If there is a regular expression R where $L(R) = A$, then there is a NFA N such that $L(N) = A$.

We can use the NFA constructions for the concatenation and kleene star as shown in the previous sections.

5.4.2 DFA to Regular Expression

Suppose A is a language over an alphabet Σ. If there is a DFA M such that $L(M) = A$, then there is a regular expression R such that $L(R) = A$.

1. Add a new start state with ϵ arrow to old start state
2. Add new accept state with ϵ arrow from old accept states. Make old accept states non-accept.
3. Remove one (of the old) states at a time: modify regular expressions on arrows that went through the removed state to restore language recognized by machine

6 Pumping Lemma

6.1 Countability

Set	Cardinality
$\{0,1\}$	2
$\{0,1\}^*$	Countably Infinite
$\mathcal{P}(\{0,1\})$	4
All languages over $\{0,1\}$ ($\mathcal{P}(\{0,1\})$)	Uncountable
Set of all regular expressions over $\{0,1\}$	Countably Infinite
Set of all regular labguages over $\{0,1\}$	Countably Infinite

Note: The power set of a countably infinite set is uncountable

6.2 Pumping Lemma Definition

If A is a regular language, then there is a number p (a pumping length) where, if s is any string in A of length at least p, then s may be divided into three pieces $s = xyz$ such that

- $|y| > 0$
- for each $i \geq 0, xy^iz \in A$
- $|xy| \leq p$

Example: A pumping length for $A = \{0,1\}^*$ is $p = 5$
$S \in A$ with $|S| \geq 5$
$x = \epsilon$ $\quad$ $y = S_1$ (the first character in S) $\quad$ z is the rest of S
$xy^iz \in \{0,1\}^*$ for all $i \geq 0$

7 Proving Nonregularity

The pumping lemma can't be used to prove that a language is regular, but it can be used to prove that a language is not regular
Proof Strategy to show that a language L is not regular

- Consider an arbitrary positive integer p
- Prove that p is not a pumping length for L
- Conclude that L does not have any pumping lengthand therefore is not regular

7.1 Examples

1. $\Sigma = \{0,1\}, L = \{0^n1^n | n \geq 0\}$
   Pick $s = 0^p1^p$
   Suppose $s = xyz$ with $|xy| \leq p$ and $|y| > 0$ where $x = 0^k, y = 0^r, z = 0^{p-k-r}1^p$
   When $i = 0, xy^iz = 0^k0^{p-k-r}1^p$ which is not apart of the language and therefore L doesn't have a pumping length.

2. $\Sigma = \{0,1\}, L = \{0^j 1^k | j \geq k \geq 0\}$
   Suppose $s = xyz$ with $|xy| \leq p$ and $|y| > 0$ where $x = 0^k, y = 0^r, z = 0^{p-k-r}1^p$
   When $i = 0, xy^i z = 0^k 0^{p-k-r} 1^p$ which is not apart of the language since less 0s than 1s and therefore L doesn't have a pumping length.

8 Pushdown Automata

A pushdown automata is like an NFA with access to a stack. At each step transition to new state, read the top of the stack, and possibly push or pop a letter from the stack.

8.1 Definition

A PDA is specified by a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$

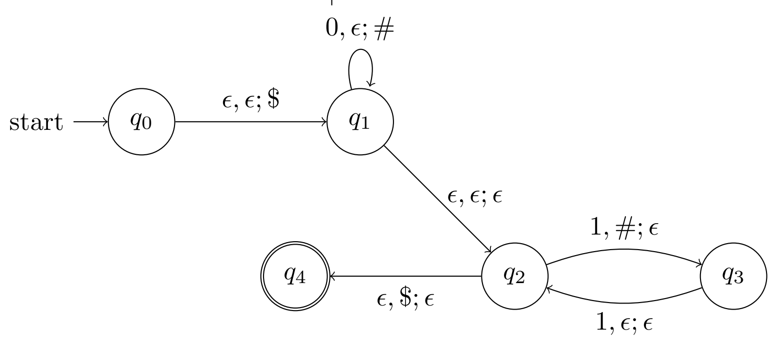
- Q is the finite set of states
- Σ is the input alphabet
- Γ is the stack alphabet
- $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow \mathcal{P}(Q \times \Gamma_\epsilon)$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states

Transition Label For PDA
input character to read or ϵ, top character of the stack to pop or ϵ, what character to push on top of the stack or ϵ

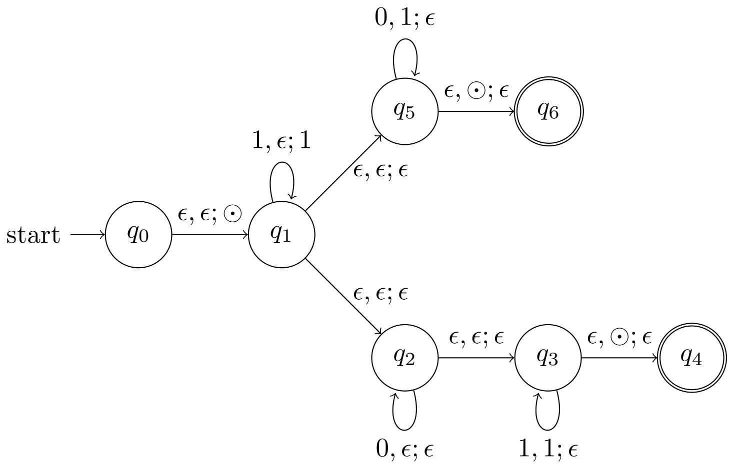
8.2 Examples

1. Language: $\{0^n 1^{2n} | n \geq 0\}$
   PDA: $(\{q_0, q_1, q_2, q_3, q_4\}, \{0,1\}, \{\$, \#\}, \delta, q_0, \{q_4\})$

	input	output
	$(q_0, 0, \$)$	$\emptyset$
δ	$\vdots$	$\vdots$
	$(q_0, \epsilon, \epsilon)$	$\{(q_1, \$)\}$
	$\vdots$	$\vdots$



2. Language: $\{1^n 0^n 1^m | n, m \geq 0\} \cup \{1^n 0^m 1^n | n, m \geq 0\}$



For each language L over Σ, there is an NFA N with $L(N) = L$ then there is a PDA M with $L(M) = L$

9 Context-free Grammars and Languages

9.1 Context-free Grammar

$G = (V, \Sigma, R, S)$

- V is a finite set of symbols that represent phases in the production pattern
- Σ is the alphabet of symbols of strings generated by CFG
- R is the set of rules
- S is the start variable

Example: $G_1 = (\{s\}, \{0\}, R, S)$ with rules

$$S \to 0S$$
$$S \to 0$$

$L(G_1) = 0^+$

9.2 Context-free Language

A language that is generated by some context-free grammar is a context-free language.

A language is generated by some context-free grammar if and only if it is recognizable by some push-down automaton.

Every Regular Language is Context Free

10 Language Closure

True/False	Closure Claim
True	The class of regular languages over Σ is closed under complementation
True	The class of regular languages over Σ is closed under union
True	The class of regular languages over Σ is closed under intersection
True	The class of regular languages over Σ is closed under concatenation
True	The class of regular languages over Σ is closed under Kleene star
False	The class of context-free languages over Σ is closed under complementation
True	The class of context-free languages over Σ is closed under union
False	The class of context-free languages over Σ is closed under intersection
True	The class of context-free languages over Σ is closed un concatenation
True	The class of context-free languages over Σ is closed under Kleene star