CS3231 Theory of Computation
AY24/25 Semester 1
by Isaac Lai

Finite Automata

DFAs

Definition. *A DFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where*

- *Q is a finite set of states*
- *Σ is a finite set of input symbols*
- *$\delta(q, w)$ is a transition function that takes as input $q \in Q$ and $w \in \Sigma$*
- *$q_0 \in Q$ is a starting state*
- *$F \subseteq Q$ is the set of accepting states*

Properties

- We can extend δ as $\hat{\delta}$ which accepts strings. Basis: $\hat{\delta}(q, \epsilon) = q$, Induction: $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$
- The **language** accepted by DFA A is $L(A) = \{w \mid \hat{\delta}(q_0, w) \in F\}$
- q is a **dead state** if $\forall w \in \Sigma^*$, $\hat{\delta}(q, w) \notin F$, i.e. an accepting state cannot be reached from q
- q is an **unreachable state** if $\forall w \in \Sigma^*$, $\hat{\delta}(q_0, w) \neq q$, i.e. q cannot be reached from the starting state

NFAs

Definition. *An NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ defined similarly to a DFA except δ maps the input (state, symbol) to a set of states (i.e. a subset of Q)*

Properties

- We can extend δ as $\hat{\delta}$ which accepts strings. Basis: $\hat{\delta}(q, \epsilon) = \{q\}$, Induction: $\hat{\delta}(q, xa) = \bigcup_{p \in \hat{\delta}(q,x)} \delta(p, a)$
- The **language** accepted by an NFA is $L(A) = \{w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$
- **NFA with ϵ-transitions**: δ maps $Q \times (\Sigma \cup \{\epsilon\})$ to subsets of Q

Definition (ϵ closures)**.** *1. $q \in Eclose(q)$*

2. If state $p \in Eclose(q)$, then each state in $\delta(p, \epsilon)$ is in $Eclose(q)$

3. Iterate step 2, until no more changes can be done to $Eclose(q)$

- For ϵ-NFA, we can extend δ to $\hat{\delta}$ where $\hat{\delta}(q, \epsilon) = Eclose(q)$ and $\hat{\delta}(q, wa) = \bigcup_{p \in R} Eclose(p)$

Equivalence

- NFAs (and ϵ-NFAs) and DFAs are **equivalent**
- To convert an NFA to a DFA, create one state in the DFA for every subset of states in the NFA. Draw the transitions accordingly from the NFA for each state in the DFA (which is a subset of the original states)

Minimisation of DFA

Suppose we have a DFA $A = (Q, \Sigma, \delta, q_0, F)$. States p, q are indistinguishable iff for all w, $\hat{\delta}(p, w) \in F$ iff $\hat{\delta}(q, w) \in F$. To build a table to determine distinguishable pairs,

1. Base case: Initially, each (p, q) pair such that $p \in F$ and $q \notin F$ (or vice versa) is distinguishable
2. Inductive step: For any $a \in \Sigma$, if $\delta(p, a)$ and $\delta(q, a)$ are distinguishable, then (p, q) are distinguishable
3. Continue the inductive step, till no more pairs of distinguishable states can be added

The new DFA is formed by

1. First delete all non-reachable states
2. Find all nondistinguishable pairs of states
3. Each pair of non-distinguishable states is equivalent, and gives an equivalence relation
4. States of the new DFA are these equivalence classes
5. If $\delta(p, a) = q$ in the original DFA, then $\delta_{new}(E_p, a) = Eq$ where E_p and E_q are equivalence classes corresponding to p and q respectively
6. Initial state of the new DFA is the equivalence class containing the start state of the original DFA, final states of the new DFA are all equivalence classes containing a final state

Parallel Simulation of 2 DFAs

- Suppose we have $A = (Q, \Sigma, \delta, q_0, F)$ and $A' = (Q', \Sigma, \delta', q_0', F')$
- Let $A'' = (Q \times Q', \Sigma, \delta', (q_0, q_0'), F'')$ where $\delta''((q, q'), a) = (\delta(q, a), \delta'(q', a))$ and F'' depends on the need. Then A'' simulates A and A' in parallel
- If $F'' = F \times F'$, then A'' accepts the intersection of languages accepted by A and A'
- If $F'' = F \times Q' \cup Q \times F'$, A'' accepts the union of languages accepted by A and A'

Regular Languages

Basis

- ϵ and $\emptyset$ are regular expressions, $L(\epsilon) = \{\epsilon\}$ and $L(\emptyset) = \emptyset$
- If $a \in \Sigma$, then a is a regular expression, and $L(a) = \{a\}$

Induction

If r_1, r_2 are regular expressions, then so are

- $r_1 + r_2$. The language is $L(r_1 + r_2) = L(r_1) \cup L(r_2)$
- $r_1 \cdot r_2$. The language is $L(r_1 \cdot r_2) = \{xy \mid x \in L(r_1), y \in L(r_2)\}$
- r_1^*. The language is $L(r_1^*) = \{x_1 x_2, \dots, x_k \mid 1 \leq i \leq k, x_i \in L(r_1)\}$
- (r_1). The language is $L((r_1)) = L(r_1)$

DFA to Regex

- $R_{i,j}^k$ denotes the regex of strings which can be formed by going from state i to j using intermediate states $\leq k$
- Base case: for $R_{i,j}^0$
  - If $i \neq j$, $R_{i,j}^0 = a_1 + a_2 + \cdots + a_m$ where $a_1, \dots, a_m$ are all symbols such that $\delta(i, a_r) = j$
  - If $i = j$, $R_{i,i}^0 = \epsilon + a_1 + \cdots + a_m$ where $a_1, \dots, a_m$ are all symbols such that $\delta(i, a_r) = i$
- Inductive case: $R_{i,j}^{k+1} = R_{i,j}^k + R_{i,k+1}^k (R_{k+1,k+1})^* R_{k+1,j}^k$
- Regex for the DFA is $\sum_{j \in F} R_{1,j}^n$

Regex Properties

- Operator precedence: $*, \cdot, +$
- $M + N = N + M$
- $L(M + N) = LM + LN$
- $L + L = L$
- $(L^*)^* = L^*$
- $\emptyset^* = \epsilon$
- $\epsilon^* = \epsilon$
- $L^+ = LL^* = L^*L$
- $L^* = \epsilon + L^+$
- $(L + M)^* = (L^* M^*)^*$

Regular Language Properties

Theorem (Pumping Lemma)**.** *Let L be a regular language. Then there exists a constant n (dependent on L) such that for every string $w \in L$ satisfying $|w| \geq n$, w can be broken into three strings $w = xyz$, such that*

1. $y \neq \epsilon$

2. $|xy| \leq n$

3. For all $k \geq 0$, $xy^k z \in L$

- If L_1, L_2 are regular, so is $L_1 \cup L_2$
- If L_1, L_2 are regular, so is $L_1 \cdot L_2$
- If L is regular, so is $\overline{L} = \Sigma^* - L$
- If L_1, L_2 are regular, so is $L_1 \cap L_2$
- If L_1, L_2 are regular, so is $L_1 - L_2$
- If L is regular, so is L^R
- Let h be a homomorphism. If L is regular, so is $h(L)$

Definition (Homomorphism)**.** *Let Σ and Γ be two alphabets, and suppose h is a mapping from Σ to Γ^*. h can be extended to strings as follows:*

- *$h(\epsilon) = \epsilon$*
- *$h(aw) = h(a) \cdot h(w)$ for any $a \in \Sigma$, $w \in \Sigma^*$*