CS 310: Automata Theory

Krishna. S

DFA Equivalence and Minimization

- Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA
- Let $L(A, q) = \{w \in \Sigma^* \mid \hat{\delta}(q, w) \in F\}$ (recall that $\hat{\delta}$ is the extended transition function, $\hat{\delta} : Q \times \Sigma^* \to Q$)
- $L(A) = L(A, q_0)$

DFA Equivalence and Minimization

- Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA
- Let $L(A, q) = \{w \in \Sigma^* \mid \hat{\delta}(q, w) \in F\}$ (recall that $\hat{\delta}$ is the extended transition function, $\hat{\delta} : Q \times \Sigma^* \to Q$)
- $L(A) = L(A, q_0)$
- Two states q_1, q_2 in A are equivalent if $L(A, q_1) = L(A, q_2)$. A state q_1 in DFA A_1 is equivalent to state q_2 in DFA A_2 if $L(A_1, q_1) = L(A_2, q_2)$.
- Two DFAs A_1, A_2 are equivalent if $L(A_1) = L(A_2)$

DFA Equivalence and Minimization

- Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA
- Let $L(A, q) = \{w \in \Sigma^* \mid \hat{\delta}(q, w) \in F\}$ (recall that $\hat{\delta}$ is the extended transition function, $\hat{\delta} : Q \times \Sigma^* \to Q$)
- $L(A) = L(A, q_0)$
- Two states q_1, q_2 in A are equivalent if $L(A, q_1) = L(A, q_2)$. A state q_1 in DFA A_1 is equivalent to state q_2 in DFA A_2 if $L(A_1, q_1) = L(A_2, q_2)$.
- Two DFAs A_1, A_2 are equivalent if $L(A_1) = L(A_2)$

DFA Equivalence

For every DFA, there exists a unique (upto state naming) minimal DFA.

Minimizing DFAs

Two observations:

- Unreachable states can be removed. This does not change the language accepted.
- Merging equivalent states. This also does not change the language accepted.

Minimizing DFAs

Two observations:

- Unreachable states can be removed. This does not change the language accepted.
- Merging equivalent states. This also does not change the language accepted.

Algorithms:

1. BFS or DFS to identify reachable states and pruning out the rest
2. Table-filling algorithm by E.F.Moore

Table-filling Algorithm

- Two states are *distinguishable* if they are not equivalent
- Formally, states q_1, q_2 are *distinguishable* if there exists $w \in \Sigma^*$ such that exactly one of $\hat{\delta}(q_1, w)$, $\hat{\delta}(q_2, w)$ is an accepting state.

Table-filling Algorithm

- Two states are *distinguishable* if they are not equivalent
- Formally, states q_1, q_2 are *distinguishable* if there exists $w \in \Sigma^*$ such that exactly one of $\hat{\delta}(q_1, w)$, $\hat{\delta}(q_2, w)$ is an accepting state.
- Table-filling is recursive discovery of distinguishable pairs of states.

Table-filling Algorithm

- Two states are *distinguishable* if they are not equivalent
- Formally, states q_1, q_2 are *distinguishable* if there exists $w \in \Sigma^*$ such that exactly one of $\hat{\delta}(q_1, w)$, $\hat{\delta}(q_2, w)$ is an accepting state.
- Table-filling is recursive discovery of distinguishable pairs of states.
- Base case : (p, q) is distinguishable if $p \in F$ and $q \notin F$ or $p \notin F, q \in F$. (why?)

Table-filling Algorithm

- Two states are *distinguishable* if they are not equivalent
- Formally, states q_1, q_2 are *distinguishable* if there exists $w \in \Sigma^*$ such that exactly one of $\hat{\delta}(q_1, w)$, $\hat{\delta}(q_2, w)$ is an accepting state.
- Table-filling is recursive discovery of distinguishable pairs of states.
- Base case : (p, q) is distinguishable if $p \in F$ and $q \notin F$ or $p \notin F, q \in F$. (why?)
- Inductive hypothesis : (p, q) is distinguishable if $\delta(p, a)$ and $\delta(q, a)$ are distinguishable for some $a \in \Sigma$. (why?)

Table-filling Algorithm

1. Distinguishable=$\{(p, q) \mid p \in F, q \notin F\}$
2. Repeat while no new pair is added
   - for every $a \in \Sigma$
     add (p, q) to Distinguishable if $(\delta(p, a), \delta(q, a)) \in$ Distinguishable.
3. Return Distinguishable.

Example on Board

Table-filling Algorithm

1. Distinguishable=$\{(p, q) \mid p \in F, q \notin F\}$
2. Repeat while no new pair is added
   - for every $a \in \Sigma$
     add (p, q) to Distinguishable if $(\delta(p, a), \delta(q, a)) \in$ Distinguishable.
3. Return Distinguishable.

Example on Board

Correctness

1. If two states are distinguished by the table-filling algorithm then they are not equivalent (obvious).
2. If two states are not distinguished by the table-filling algorithm then they are equivalent.

Correctness

- Assume (p, q) is a pair which is not distinguished by the algorithm, but they are not equivalent.

Correctness

- Assume (p, q) is a pair which is not distinguished by the algorithm, but they are not equivalent.
- That is, (p, q) is distinguishable, but the algorithm did not find it
- Call such (p, q) a bad pair. There must be some $w \in \Sigma^*$ that distinguishes (p, q).

Correctness

- Assume (p, q) is a pair which is not distinguished by the algorithm, but they are not equivalent.
- That is, (p, q) is distinguishable, but the algorithm did not find it
- Call such (p, q) a bad pair. There must be some $w \in \Sigma^*$ that distinguishes (p, q).
- Take the shortest such distinguishing word w among all bad pairs, and consider the corresponding bad pair (p, q).

Correctness

- Assume (p, q) is a pair which is not distinguished by the algorithm, but they are not equivalent.
- That is, (p, q) is distinguishable, but the algorithm did not find it
- Call such (p, q) a bad pair. There must be some $w \in \Sigma^*$ that distinguishes (p, q).
- Take the shortest such distinguishing word w among all bad pairs, and consider the corresponding bad pair (p, q).
  - $w \neq \epsilon$ (why?)

Correctness

- Assume (p, q) is a pair which is not distinguished by the algorithm, but they are not equivalent.
- That is, (p, q) is distinguishable, but the algorithm did not find it
- Call such (p, q) a bad pair. There must be some $w \in \Sigma^*$ that distinguishes (p, q).
- Take the shortest such distinguishing word w among all bad pairs, and consider the corresponding bad pair (p, q).
  - $w \neq \epsilon$ (why?)
  - Let $w = ax$. As p, q are distinguishable, exactly one of $\hat{\delta}(p, ax), \hat{\delta}(q, ax)$ is accepting.
  - Then $p' = \delta(p, a)$ and $q' = \delta(q, a)$ are distinguished by x.

Correctness

- Assume (p, q) is a pair which is not distinguished by the algorithm, but they are not equivalent.
- That is, (p, q) is distinguishable, but the algorithm did not find it
- Call such (p, q) a bad pair. There must be some $w \in \Sigma^*$ that distinguishes (p, q).
- Take the shortest such distinguishing word w among all bad pairs, and consider the corresponding bad pair (p, q).
  - $w \neq \epsilon$ (why?)
  - Let $w = ax$. As p, q are distinguishable, exactly one of $\hat{\delta}(p, ax), \hat{\delta}(q, ax)$ is accepting.
  - Then $p' = \delta(p, a)$ and $q' = \delta(q, a)$ are distinguished by x.
  - If (p', q') was discovered by the algorithm, then (p, q) would have been discovered as well.

Correctness

- Assume (p, q) is a pair which is not distinguished by the algorithm, but they are not equivalent.
- That is, (p, q) is distinguishable, but the algorithm did not find it
- Call such (p, q) a bad pair. There must be some $w \in \Sigma^*$ that distinguishes (p, q).
- Take the shortest such distinguishing word w among all bad pairs, and consider the corresponding bad pair (p, q).
  - $w \neq \epsilon$ (why?)
  - Let $w = ax$. As p, q are distinguishable, exactly one of $\hat{\delta}(p, ax), \hat{\delta}(q, ax)$ is accepting.
  - Then $p' = \delta(p, a)$ and $q' = \delta(q, a)$ are distinguished by x.
  - If (p', q') was discovered by the algorithm, then (p, q) would have been discovered as well.
  - If (p', q') is not discovered by the algorithm, then (p', q') is a bad pair with a shorter distinguishing word, a contradiction.

Minimization of DFAs

- Let A be a DFA with no unreachable states
- Let $\approx \subseteq Q \times Q$ be the state equivalence relation (computed say by the table-filling algorithm)

$$p \approx q \Leftrightarrow \forall x \in \Sigma^* (\hat{\delta}(p, x) \in F \Leftrightarrow \hat{\delta}(q, x) \in F)$$

Minimization of DFAs

- Let A be a DFA with no unreachable states
- Let $\approx \subseteq Q \times Q$ be the state equivalence relation (computed say by the table-filling algorithm)

$$p \approx q \Leftrightarrow \forall x \in \Sigma^* (\hat{\delta}(p, x) \in F \Leftrightarrow \hat{\delta}(q, x) \in F)$$

- $\approx$ is an equivalence relation with finitely many classes
- Let $[q] = \{q' \mid q' \approx q\}$

Minimization of DFAs

- Let A be a DFA with no unreachable states
- Let $\approx \subseteq Q \times Q$ be the state equivalence relation (computed say by the table-filling algorithm)

$$p \approx q \Leftrightarrow \forall x \in \Sigma^* (\hat{\delta}(p, x) \in F \Leftrightarrow \hat{\delta}(q, x) \in F)$$

- $\approx$ is an equivalence relation with finitely many classes
- Let $[q] = \{q' \mid q' \approx q\}$
- Given DFA $A = (Q, \Sigma, \delta, q_0, F)$, we can minimize A to the DFA $A_{min} = (Q', \Sigma, \delta', q_0', F')$ called the *Quotient Automata* where
  - $Q' = \{[q] \mid q \in Q\}$
  - $\delta'([q], a) = [\delta(q, a)]$ for all $a \in \Sigma$,

Minimization of DFAs

- Let A be a DFA with no unreachable states
- Let $\approx \subseteq Q \times Q$ be the state equivalence relation (computed say by the table-filling algorithm)

$$p \approx q \Leftrightarrow \forall x \in \Sigma^* (\hat{\delta}(p, x) \in F \Leftrightarrow \hat{\delta}(q, x) \in F)$$

- $\approx$ is an equivalence relation with finitely many classes
- Let $[q] = \{q' \mid q' \approx q\}$
- Given DFA $A = (Q, \Sigma, \delta, q_0, F)$, we can minimize A to the DFA $A_{min} = (Q', \Sigma, \delta', q_0', F')$ called the *Quotient Automata* where
  - $Q' = \{[q] \mid q \in Q\}$
  - $\delta'([q], a) = [\delta(q, a)]$ for all $a \in \Sigma$,
    If $p \approx q$ then $\delta(p, a) \approx \delta(q, a)$. That is, if $[p] = [q]$, then $[\delta(p, a) = [\delta(q, a)]$
  - $q_0' = [q_0]$
  - $F' = \{[q] \mid q \in F\}$

Minimization of DFAs

- Let A be a DFA with no unreachable states
- Let $\approx \subseteq Q \times Q$ be the state equivalence relation (computed say by the table-filling algorithm)

$$p \approx q \Leftrightarrow \forall x \in \Sigma^* (\hat{\delta}(p, x) \in F \Leftrightarrow \hat{\delta}(q, x) \in F)$$

- $\approx$ is an equivalence relation with finitely many classes
- Let $[q] = \{q' \mid q' \approx q\}$
- Given DFA $A = (Q, \Sigma, \delta, q_0, F)$, we can minimize A to the DFA $A_{min} = (Q', \Sigma, \delta', q_0', F')$ called the *Quotient Automata* where
  - $Q' = \{[q] \mid q \in Q\}$
  - $\delta'([q], a) = [\delta(q, a)]$ for all $a \in \Sigma$,
    If $p \approx q$ then $\delta(p, a) \approx \delta(q, a)$. That is, if $[p] = [q]$, then $[\delta(p, a) = [\delta(q, a)]$
  - $q_0' = [q_0]$
  - $F' = \{[q] \mid q \in F\}$
    $q \in F$ iff $[q] \in F'$. If $p \approx q$ and $q \in F$, then $p \in F$. Each class is either inside F or disjoint from F.

Minimization of DFAs

$L(A_{min}) = L(A)$

1. $x \in L(A_{min})$ iff $\hat{\delta}'(q'_0, x) \in F'$ iff

Minimization of DFAs

$L(A_{min}) = L(A)$

1. $x \in L(A_{min})$ iff $\hat{\delta}'(q'_0, x) \in F'$ iff
2. $\hat{\delta}'([q_0], x) \in F'$ iff

Minimization of DFAs

$L(A_{min}) = L(A)$

1. $x \in L(A_{min})$ iff $\hat{\delta}'(q_0', x) \in F'$ iff
2. $\hat{\delta}'([q_0], x) \in F'$ iff
3. $[\hat{\delta}(q_0, x)] \in F'$ iff

Minimization of DFAs

$L(A_{min}) = L(A)$

1. $x \in L(A_{min})$ iff $\hat{\delta}'(q_0', x) \in F'$ iff
2. $\hat{\delta}'([q_0], x) \in F'$ iff
3. $[\hat{\delta}(q_0, x)] \in F'$ iff
4. $\hat{\delta}(q_0, x) \in F$ iff

Minimization of DFAs

$L(A_{min}) = L(A)$

1. $x \in L(A_{min})$ iff $\hat{\delta}'(q_0', x) \in F'$ iff
2. $\hat{\delta}'([q_0], x) \in F'$ iff
3. $[\hat{\delta}(q_0, x)] \in F'$ iff
4. $\hat{\delta}(q_0, x) \in F$ iff
5. $x \in L(A)$

Minimization of DFAs

$L(A_{min}) = L(A)$

1. $x \in L(A_{min})$ iff $\hat{\delta}'(q_0', x) \in F'$ iff
2. $\hat{\delta}'([q_0], x) \in F'$ iff
3. $[\hat{\delta}(q_0, x)] \in F'$ iff
4. $\hat{\delta}(q_0, x) \in F$ iff
5. $x \in L(A)$

Claim

1. No two distinct states in A_{min} are equivalent.
2. A_{min} is the minimum and unique (upto state renaming) DFA equivalent to A.

Minimality of A_{min}

- Assume there is a DFA B with smaller number of states than A_{min}, such that $L(B) = L(A)$.

Minimality of A_{min}

- Assume there is a DFA B with smaller number of states than A_{min}, such that $L(B) = L(A)$.
- Using the table-filling algorithm, compute equivalent states of B and A_{min}.

Minimality of A_{min}

- Assume there is a DFA B with smaller number of states than A_{min}, such that $L(B) = L(A)$.
- Using the table-filling algorithm, compute equivalent states of B and A_{min}.
- The initial states of B and A_{min} must be equivalent (why?)

Minimality of A_{min}

- Assume there is a DFA B with smaller number of states than A_{min}, such that $L(B) = L(A)$.
- Using the table-filling algorithm, compute equivalent states of B and A_{min}.
- The initial states of B and A_{min} must be equivalent (why?)
- After reading any $w \in \Sigma^*$ from their initial states, both DFAs enter equivalent states (why?)

Minimality of A_{min}

- Assume there is a DFA B with smaller number of states than A_{min}, such that $L(B) = L(A)$.
- Using the table-filling algorithm, compute equivalent states of B and A_{min}.
- The initial states of B and A_{min} must be equivalent (why?)
- After reading any $w \in \Sigma^*$ from their initial states, both DFAs enter equivalent states (why?)
- For every state of A_{min}, there is an equivalent state in B
- Since the number of states in B are $<$ than those in A_{min}, there are at least two states p, q in A_{min} equivalent to some state in B.
- That is, p, q are equivalent, a contradiction.

A minimal DFA directly from the language

Given a regular language R, can we define the minimal DFA for R directly from it?

Given a language L (not necessarily regular), and words $u, v \in L$, define $u \sim_L v$ if for all $w \in \Sigma^*$, $uw \in L \Leftrightarrow vw \in L$.

- $\sim_L \subseteq \Sigma^* \times \Sigma^*$ is an equivalence relation on words
- Consider the equivalence classes of $\sim_L$
- When can $\sim_L$ have only finitely many equivalence classes?

Properties of $\sim_L$

- $\sim_L$ is a right congruence : that is, for any $a \in \Sigma$, $x \sim_L y \Rightarrow xa \sim_L ya$
- $\sim_L$ refines L : that is, $x \sim_L y \Rightarrow x \in L \Leftrightarrow y \in L$

Myhill-Nerode relations

An equivalence relation on Σ^* is called a *Myhill-Nerode* relation for $L \subseteq \Sigma^*$ if it is a right congruence refining L, and has finitely many equivalence classes.
The relation was proposed by John Myhill and Anil Nerode.

An Example

Consider $L = \{a^n b^n \mid n \geqslant 0\}$. Recall $\sim_L$.

- Consider the set of words $\{a, a^2, \ldots, a^i, \ldots\}$.
- If $\sim_L$ has finitely many classes, then there exists a^m, a^n $m \neq n$ such that $a^m \sim_L a^n$. That is, for all $w \in \Sigma^*$, $a^m.w \in L$ iff $a^n.w \in L$.
- However, $a^m b^m \in L$ but $a^n b^m \notin L$ if we choose $w = a^m$
- Hence $[a^m] \neq [a^n]$ for $m \neq n$.
- Infinitely many equivalence classes.

An Example

Consider $L = a^* b^*$. Recall $\sim_L$. What are the classes of $\sim_L$?

An Example

Consider $L = a^*b^*$. Recall $\sim_L$. What are the classes of $\sim_L$?

- $[\epsilon] = a^*$
- $[b] = a^*b^+$
- $[ba] = a^*b^+a\Sigma^*$

Myhill-Nerode Relations

If L is regular, then there exists a Myhill-Nerode relation $\sim_A$ for L defined from the DFA A for L.

Myhill-Nerode Relations

If L is regular, then there exists a Myhill-Nerode relation $\sim_A$ for L defined from the DFA A for L.

- Let $A = (Q, \Sigma, \delta, q_0, F)$ be the DFA with no inaccessible states such that $L = L(A)$.
- Define $x \sim_A y$ iff $\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$.
- It can be seen that $\sim_A$ is a right congruence refining L.
- $[x] = \{y \mid \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)\}$. There is one equivalence class corresponding to each state in Q. Hence, finitely many classes.
- Hence, $\sim_A$ is Myhill-Nerode.

Myhill-Nerode Relations

If language L has a Myhill-Nerode relation $\sim$, then L is regular.

Myhill-Nerode Relations

If language L has a Myhill-Nerode relation $\sim$, then L is regular.

- We know $\sim$ is a right congruence refining L, with finitely many equivalence classes $[x]$.

Myhill-Nerode Relations

If language L has a Myhill-Nerode relation $\sim$, then L is regular.

- We know $\sim$ is a right congruence refining L, with finitely many equivalence classes $[x]$.
- From $\sim$, define a DFA $A_\sim = (Q, \Sigma, \delta, s, F)$ where
  $Q = \{[x] \mid x \in \Sigma^*\}$
  $s = [\epsilon]$
  $F = \{[x] \mid x \in L\}$
  $\delta([x], a) = [xa]$.
- Is $A_\sim$ well-defined?
- $x \in L$ iff $[x] \in F$. If $[x] \in F$, then all words in $[x]$ are in L as $\sim$ refines L.
- $L(A_\sim) = L$. $x \in L(A_\sim)$ iff $\hat{\delta}([\epsilon], x) \in F$ iff $[x] \in F$ iff $x \in L$.

Two constructions

We did two things.

(1) Given a DFA A accepting L with no inaccessible state, we defined a Myhill-Nerode relation $\sim_A$ from A.

(2) Given a language L with a Myhill-Nerode relation $\sim$, we constructed the DFA $A_\sim$ for L.

(1), (2) are inverses upto isomorphism. That is,

- Myhill-Nerode relation $\sim \to$ DFA $A_\sim \to$ Myhill-Nerode relation $\sim_{A_\sim}$ would mean $\sim = \sim_{A_\sim}$.
- DFA A for language L to Myhill-Nerode relation $\sim_A$ to DFA $A_{\sim_A}$ would imply A is isomorphic to $A_{\sim_A}$.

Myhill-Nerode and the Minimal DFA

- A relation $\sim_1$ is said to refine another relation $\sim_2$ if $\sim_1 \subseteq \sim_2$ when considered as sets of ordered pairs.
- That is, $x \sim_1 y \Rightarrow x \sim_2 y$.
- In other words, $[x]_1 \subseteq [x]_2$

Myhill-Nerode and the Minimal DFA

- A relation $\sim_1$ is said to refine another relation $\sim_2$ if $\sim_1 \subseteq \sim_2$ when considered as sets of ordered pairs.
- That is, $x \sim_1 y \Rightarrow x \sim_2 y$.
- In other words, $[x]_1 \subseteq [x]_2$
- The Myhill-Nerode relation $\sim_L$ for a language L refines the relation which has 2 classes L and $\Sigma^* \setminus L$.
- If $\sim_1$ refines $\sim_2$, then $\sim_2$ is coarser than $\sim_1$ while $\sim_1$ is finer than $\sim_2$
- Any set U, has a finest and coarsest equivalence relation : finest is the identity and coarsest is universal relation $\{(x, y) \mid x, y \in U\}$.

Myhill-Nerode and the Minimal DFA

Let L be some language. Recall the relation $\sim_L$ we defined on L. $\sim_L$ is a right congruence refining L and is the coarsest such relation.

- We already know $\sim_L$ is a right congruence refining L.
- Show that $\sim_L$ is coarsest. Let $\equiv$ be a right congruence on L refining L. Then $\equiv$ refines $\sim_L$:
  - $x \equiv y \Rightarrow \forall z(xz \equiv yz)$ (why?)
  - $\forall z(xz \equiv yz) \Rightarrow \forall z(xz \in L \Leftrightarrow yz \in L)$ (why?)
  - $\forall z(xz \in L \Leftrightarrow yz \in L) \Rightarrow x \sim_L y$ (definition of $\sim_L$)

Myhill-Nerode Theorem

The following are equivalent :

1. L is regular
2. there exists a Myhill-Nerode relation $\sim_L$ for L
3. $\sim_L$ is of finite index, that is, has finitely many equivalence classes.

Since $\sim_L$ is the coarsest, the DFA it produces for L has the fewest states among all DFAs for L.

The table-filling algorithm gives the above minimal DFA. Show that the relation $\sim_A$ computed from the collapsed DFA A is same as $\sim_L$.

Table-filling and the Myhill-Nerode DFA

Assume $A = (Q, \Sigma, \delta, q_0, F)$ is a DFA for L which has been collapsed by the table-filling algorithm. Recall the equivalence computed by the table-filling algorithm

$$p \approx q \Leftrightarrow \forall x \in \Sigma^* (\hat{\delta}(p, x) \in F \Leftrightarrow \hat{\delta}(q, x) \in F)$$

- If A is the collapsed automaton produced for L by the table-filling algorithm, we show that $\sim_L$ and $\sim_A$ are the same, where $\sim_A$ is the Myhill-Nerode relation constructed from A.
- $x \sim_L y$ iff $\forall z(xz \in L \Leftrightarrow yz \in L)$
- $\Leftrightarrow \forall z(\hat{\delta}(q_0, xz) \in F \Leftrightarrow \hat{\delta}(q_0, yz) \in F))$
- $\Leftrightarrow \hat{\delta}(q_0, x) \approx \hat{\delta}(q_0, y)$
- $\Leftrightarrow \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$ as A is collapsed
- $\Leftrightarrow x \sim_A y$