Equivalence Classes

Consider any regular language L.

- $u \equiv_L w$ iff for all x, $ux \in L$ iff $wx \in L$.
- Note that $\equiv_L$ is indeed an equivalence relation, as it is reflexive, symmetric and transitive.
- Let $equiv(w)$ denote the equivalence class of w.
- If the number of equivalence classes as above is finite, then form a DFA $(Q, \Sigma, \delta, q_0, F)$ as follows:
  $Q = \{equiv(w) : w \in \Sigma^*\}$.
  $q_0 = equiv(\epsilon)$.
  $F = \{equiv(w) : w \in L\}$.
  $\delta(equiv(w), a) = equiv(wa)$.
- Note that δ is well defined as $u \equiv_L w$ implies $ua \equiv_L wa$.

The method in previous slide gives the minimal DFA for L.
Why cannot there be a smaller DFA accepting the same language?
Suppose $A' = (Q', \Sigma, \delta', q_0', F')$ is an automata accepting L.

Claim: $u \not\equiv_L w$ implies $\hat{\delta}'(q_0', u) \neq \hat{\delta}'(q_0', w)$.
Proof: Suppose otherwise, that is $u \not\equiv_L w$ but $\hat{\delta}'(q_0', u) = \hat{\delta}'(q_0', w)$.
Then, there exists a string x such that $ux \in L$ but $wx \notin L$ (or vice versa), by definition of $\equiv_L$.
But $\hat{\delta}'(q_0', ux) = \hat{\delta}'(q_0', wx)$ and thus A' accepts both ux and wx or accepts none of ux, wx. A contradiction.

Let $u \equiv_{A'} w$ iff $\hat{\delta}'(q_0', u) = \hat{\delta}'(q_0', w)$.
By above claim it follows that $\equiv_{A'}$ divides the equivalence classes $\equiv_L$ into finer equivalence classes.
Thus, the DFA given using $\equiv_L$ is minimal. Furthermore, it is same as any other minimal automata except for renaming of the states.
Thus, the method given above gives minimal DFA, and it is unique!

We will use the above result for finding a minimal equivalent DFA for a language L, from a given DFA $A = (Q, \Sigma, \delta, q_0, F)$ for L.
Note that if $\hat{\delta}(q_0, u) = \hat{\delta}(q_0, v)$, then $u \equiv_L v$.
However, it is possible that there are further equivalences: that is it may be true that $\hat{\delta}(q_0, u) \neq \hat{\delta}(q_0, v)$, but $u \equiv_L v$.
The aim of next algorithm is to find such u, v and then merge the states $q = \hat{\delta}(q_0, u)$ and $q' = \hat{\delta}(q_0, v)$.

Minimization of Automata; Equivalence

Suppose we are given $A = (Q, \Sigma, \delta, q_0, F)$.

(a) We say that (p, q) are distinguishable iff there exists a string w such that either

$$\hat{\delta}(p, w) \in F,\ \hat{\delta}(q, w) \notin F, \text{ or}$$
$$\hat{\delta}(p, w) \notin F,\ \hat{\delta}(q, w) \in F.$$

(b) In other words, (p, q) are indistinguisable iff for all w, $\hat{\delta}(p, w) \in F$ iff $\hat{\delta}(q, w) \in F$.

Table building algorithm for determining all pairs that are distinguishable.
1. Base Case: Initially, each pair (p, q) 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, until it can add no more pairs of distinguishable states.
Then the remaining pairs are nondistinguishable states.

Form a new DFA as follows:
0. First delete all non-reachable states.
1. Find all nondistinguishable pairs of states.
2. Each pair of non-distinguishable states is equivalent, and it gives an equivalence relation.
3(a). States of the new DFA are these equivalence classes.
3(b). Transition from each equivalence class above on input a is based on the corresponding transition in original DFA, i.e., if $\delta(p, a) = q$ in the original automata, then $\delta_{new}(Ep, a) = Eq$, where Ep and Eq are equivalence classes corresponding to p and q respectively.
3(c). Initial state of the new automata is the equivalence class containing the starting state of original automata, and final states of the new automata are all the equivalence classes containing a final state.

Why does above work?

- For automata A accepting L, let $u \equiv_A w$ iff $\hat{\delta}(q_0, u) = \hat{\delta}(q_0, w)$.
- $u \equiv_A w$ implies $u \equiv_L w$, as for all x, $ux \in L(A) = L$ iff $wx \in L(A) = L$.
- Thus, we are looking for which $\equiv_A$ classes are to be merged together to form $\equiv_L$.
- Distinguishablity of p, q thus tells us that p, q cannot be merged together.
- Indistinguishablity of p, q tells us that p, q can be merged together.

Why does the algorithm find all and only pairs of distinguishable states?

By induction on number of steps, if our induction algorithm says two states p, q are distinguishable, then they are distinguishable.
Base case: ϵ distinguishes the accepting and non-accepting states.
Induction step: Suppose we add (p, q) to distinguishable state pairs, due to $(\delta(p, a) = p'$, $\delta(q, a) = q')$ being distinguishable.

Then as (p', q') are distinguishable, for some x, $\hat{\delta}(p', x) \in F$ and $\hat{\delta}(q', x) \notin F$ (or vice versa).

But then, $\hat{\delta}(p, ax) \in F$ and $\hat{\delta}(q, ax) \notin F$ (or vice versa).
Thus, (p, q) are distinguishable.

Algorithm finds all pairs of distinguishable states.
By induction on the length of strings which distinguish the states.
Base case: Clearly, the algorithm finds all pairs of states which can be distinguished using strings of length 0 (that is by using ϵ).
Induction: Suppose the algorithm finds all pairs of states which can be distinguished using strings of length at most k.
Then, consider any pair of states (p, q) which can be distinguished using string $w = ax$ of length $k + 1$.
Then, since the algorithm finds the pair $(\delta(p, a), \delta(q, a))$ as distinguishable (by induction), we have that the algorithm in the induction step will find that (p, q) are distinguishable.