CS310 : Automata Theory 2019

Lecture 12: Equivalence

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2019-02-01

Are two descriptions of regular languages

equivalent,

i.e., describing same language?

Unique DFA

For starters, we covert all descriptions to DFAs.

DFA representation does not guarantee uniqueness.

Can we transform the DFAs to canonical forms to compare their languages?

We will proceed by learning to identify equivalent states in DFAs.

Equivalent states

Definition 12.1

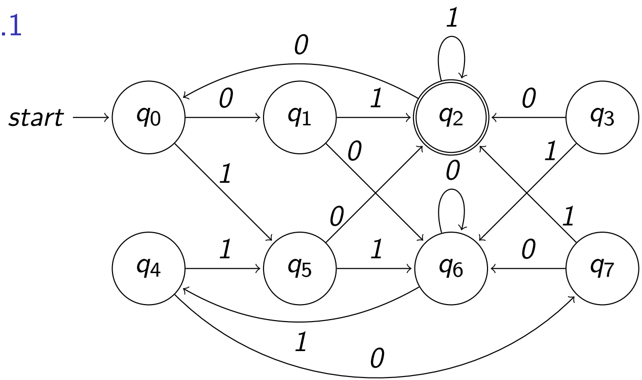
Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA. $p, q \in Q$ are equivalent *if for each word w, $\hat{\delta}(p, w)$ is accepting state iff $\hat{\delta}(q, w)$ is accepting state.*

Definition 12.2

We say p and q are distinguishable *if they are not equivalent, that is, there is a word w such that one of $\hat{\delta}(p, w)$ and $\hat{\delta}(q, w)$ is accepting but the other is not.*

Example: distinguishable states

Example 12.1



- ϵ *distinguishes* q_2 *and* q_6.
- 01 *distinguishes* q_0 *and* q_6.

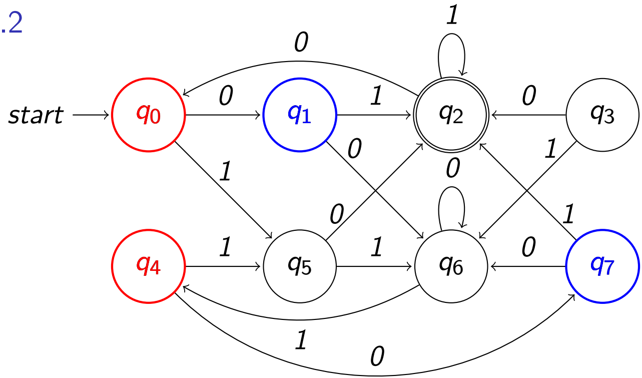
Exercise 12.1

Give strings that distinguishes the following pair of states.

- q_1 *and* q_5
- q_2 *and* q_7
- q_4 *and* q_3
- q_1 *and* q_7

Example : equivalent states

Example 12.2



- q_1 and q_7 are equivalent
- q_0 and q_4 are equivalent

Exercise 12.2

Are there any other equivalent pairs?

Identifying distinguishable pairs

base case:
It is clear that accepting states are distinguishable from the others.

induction step:
We can also declare a pair of states distinguishable if they take us to already known distinguishable pair of states upon reading some letter.

Example 12.3

Let us suppose word w distinguishes states q and p.

Let $\delta(r, a) = q$ and $\delta(s, a) = p$.

Therefore, word aw distinguishes r and s.

Algorithm for identifying distinguishable pairs

Algorithm 12.1: DistinguishablePairs(DFA $A = (Q, \Sigma, \delta, q_0, F)$)

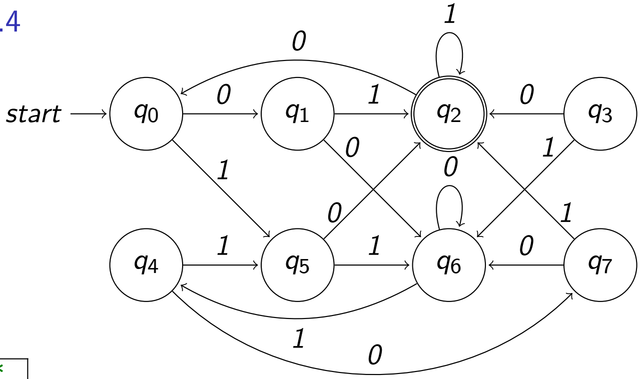
Output: set of unordered pairs of states D

$D := \{\{q, q'\} | q \in F \wedge q' \in Q - F\}$;
changed := *true*;
while *changed* **do**
    changed := *false*;
    if $\exists q, q', a$ *such that* $\{q, q'\} \notin D$ *and* $\{\delta(q, a), \delta(q', a)\} \in D$ **then**
        $D := D \cup \{ \{q, q'\} \}$;
        changed := *true*;
return D

Since D can be viewed as yes/no entries in a table, the above is also called table filling algorithm.

Example: run of DistinguishablePairs

Example 12.4



q_1	*						
q_2	*	*					
q_3	*	*	*				
q_4		*	*	*			
q_5	*	*	*		*		
q_6	*	*	*	*	*	*	
q_7	*		*	*	*	*	*
D	q_0	q_1	q_2	q_3	q_4	q_5	q_6

Distinguishing words

- ϵ
- 0
- 1
- 01

Finding equivalent states

Theorem 12.1

Let $D = \text{DistinguishablePairs}((Q, \Sigma, \delta, q_0, F))$.
If $\{q, q'\} \notin D$, *then* q *and* q' *are equivalent.*

Proof.

Let $D' = \{\{q, q'\} | q \text{ and } q' \text{ are not equivalent, and } \{q, q'\} \notin D\}$

We choose the shortest distinguishing word w for some $\{q, q'\} \in D'$. (why exists?)

$w \neq \epsilon$ is not possible, since the initialization of D adds all the pairs due to ϵ.

Let $w = ax$. $\{\delta(q, a), \delta(q', a)\}$ should be distinguishable by x.

If $\{\delta(q, a), \delta(q', a)\}$ is in D, then $\{q, q'\}$ must be in D.

Otherwise, $\{\delta(q, a), \delta(q', a)\}$ has shorter distinguishing word.
Contradiction. □

Runtime performance of DistinguishablePairs

Let us assume *DFA* has n states and constant number of alphabets.

- There can be at most n^2 iterations
- In each iteration, we search for q, q', and a, which may take n^2 steps to find the states.

Total runtime n^4.

Better implementation of DistinguishablePairs

We can improve the performance by preprocessing.

Create the following directed graph

- nodes are the unordered pairs $\{q, q'\}$
- edges are $(\{q, q'\}, \{s, s'\})$ if $\delta(s, a) = q$ and $\delta(s', a) = q'$ for some a.

All nodes reachable from $\{\{q, q'\} | q \in F \wedge q' \in Q - F\}$ are distinguishable pairs.

We can construct the graph in flight.

Exercise 12.3

What is the complexity after the above preprocessing?

Topic 12.1

RL equivalence checking

Regular language equivalence checking

Consider two regular languages L_1 and L_2.

Convert them into DFA A_1 and A_2 respectively.

Run DistinguishablePairs on the automata as if they were one.

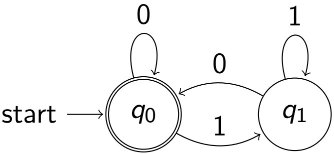
Theorem 12.2

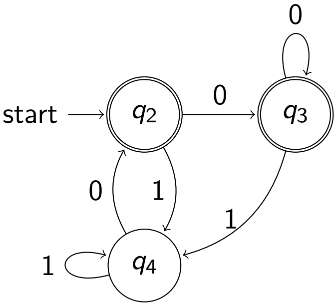
The initial states of A_1 and A_2 are equivalent iff $L_1 = L_2$.

Example : language equivalence checking

Example 12.5

Run DistinguishablePairs *on the following automata as if they are one.*





We obtain the following D set

q_1	*			
q_2		*		
q_3		*		
q_4	*		*	*
D	q_0	q_1	q_2	q_3

Since q_0 and q_2 are not distinguishable, the automata have same language.

End of Lecture 12