



CSCI 742 - Compiler Construction

Lecture 7

Building Efficient Lexers

Instructor: Hossein Hojjat

January 31, 2017

Lexer Automatic Construction: Big Picture

Input: Token Spec

- List of regular expressions (RE) in priority order

Output: Lexer

- Reads an input stream and breaks it up into tokens according to REs

Algorithm

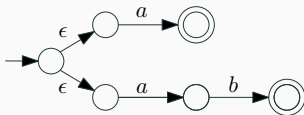
- Convert REs into non-deterministic finite automata (NFA)
- Convert NFA to DFA
- Convert DFA into transition table

Lexer Automatic Construction: Example

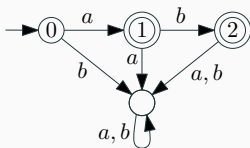
- RE for tokens:

$(a|ab)$

- NFA:



- DFA:



- Transition Table:

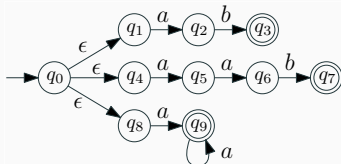
	a	b
0	1	Error
1	Error	2
2	Error	Error

Lexer Automatic Construction: Example

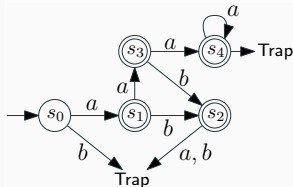
Token Specification

ab {Action 1}
 aab {Action 2}
 $a+$ {Action 3}

NFA



DFA



	a	b
s_0	s_1	Error
s_1	s_3	s_2
s_2	Error	Error
s_3	s_4	s_2
s_4	s_4	Error

$\Sigma = \{a, b\}$

Example:

Input: aab

- $s_0 \rightarrow s_1 \rightarrow s_3 \rightarrow s_2$

Theorem

A language L can be described by regular expression if and only if L is the language accepted by a finite automaton.

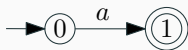
Algorithms:

- Regular expression \Rightarrow Automaton
 - important for lexer construction
- Automaton \Rightarrow Regular expression
 - interesting method in formal languages theory

RE \Rightarrow Finite Automaton

- Build the finite automaton inductively, based on the definition of regular expressions

a



ϵ

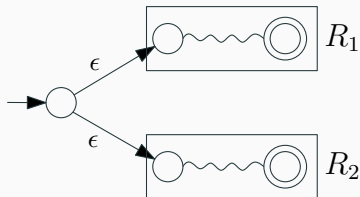


\emptyset

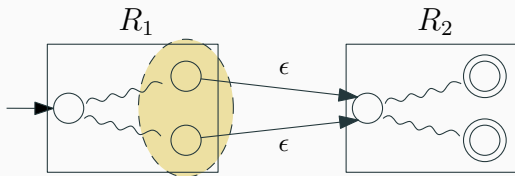


RE \Rightarrow Finite Automaton

Alternation $R_1 \mid R_2$

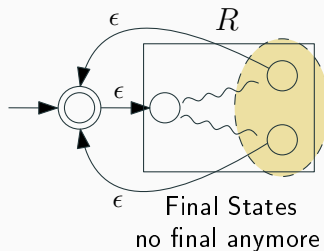


Concatenation
 $R_1 \cdot R_2$



Final States
no final anymore

Alternation R^*



Question

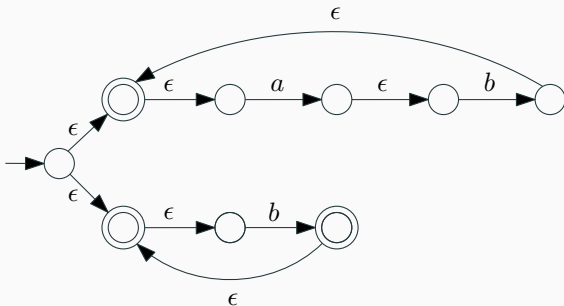
- Construct an NFA for the regular expression $(ab)^* \mid b^*$

Exercise

Question

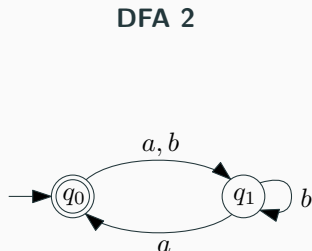
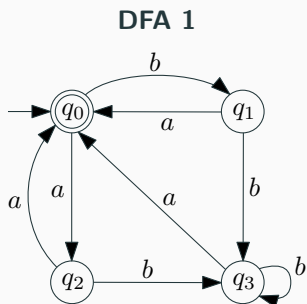
- Construct an NFA for the regular expression $(ab)^* \mid b^*$

Answer



DFA Minimization

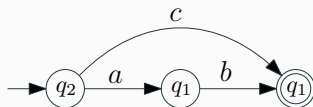
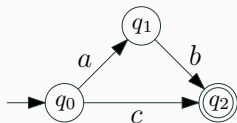
- Generated DFAs may have a large number of states
- **DFA Minimization:** Converts a DFA to another DFA that:
 - recognizes the same language
 - has a minimum number of states
- Increases time/space efficiency



Both DFAs accept: $((a \mid b)b^*a)^*$

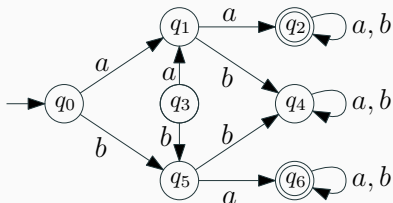
DFA Minimization

- For every regular language L there exists a unique minimal DFA that recognizes L
 - uniqueness up to renaming of states (isomorphism)
- Minimal DFA can be found mechanically



DFA Minimization: Example

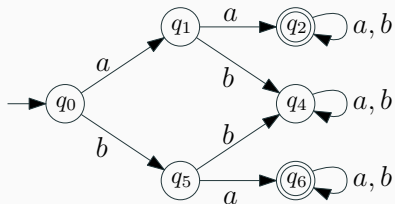
- Remove unreachable states: there is no path from initial state to q_3



$$\Sigma = \{a, b\}$$

DFA Minimization: Example

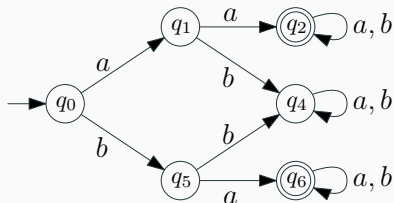
- Remove unreachable states: there is no path from initial state to q_3



$$\Sigma = \{a, b\}$$

DFA Minimization: Example

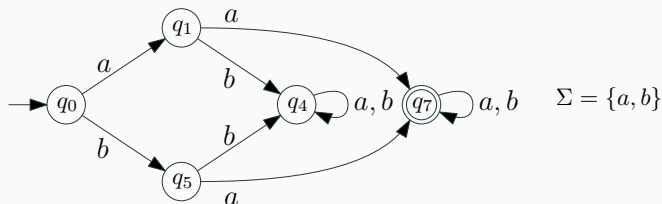
- Remove unreachable states: there is no path from initial state to q_3
- q_2 , q_6 are both accepting sinks with self-loop for any character in Σ
- Any string reaches q_2 or q_6 is guaranteed to be accepted later
- q_2 and q_6 are **equivalent** states: we can unify them



$$\Sigma = \{a, b\}$$

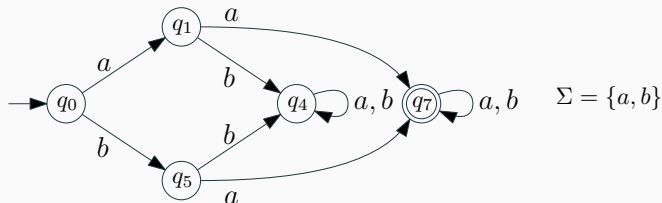
DFA Minimization: Example

- Remove unreachable states: there is no path from initial state to q_3
- q_2 , q_6 are both accepting sinks with self-loop for any character in Σ
- Any string reaches q_2 or q_6 is guaranteed to be accepted later
- q_2 and q_6 are **equivalent** states: we can unify them



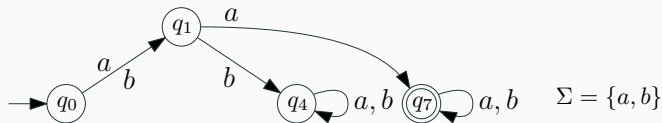
DFA Minimization: Example

- Remove unreachable states: there is no path from initial state to q_3
- q_2 , q_6 are both accepting sinks with self-loop for any character in Σ
- Any string reaches q_2 or q_6 is guaranteed to be accepted later
- q_2 and q_6 are **equivalent** states: we can unify them
- If DFA is in q_1 or q_5 :
 - if next character is a , it forever accepts in both states
 - if next character is b , it forever rejects in both states
- q_1 and q_5 are **equivalent** states: we can unify them



DFA Minimization: Example

- Remove unreachable states: there is no path from initial state to q_3
- q_2 , q_6 are both accepting sinks with self-loop for any character in Σ
- Any string reaches q_2 or q_6 is guaranteed to be accepted later
- q_2 and q_6 are **equivalent** states: we can unify them
- If DFA is in q_1 or q_5 :
 - if next character is a , it forever accepts in both states
 - if next character is b , it forever rejects in both states
- q_1 and q_5 are **equivalent** states: we can unify them



Equivalent States

Intuition

- Two states are equivalent if all subsequent behavior from those states is the same
- Equivalent states may be unified without affecting DFA's behavior

Definition

- We say that states p and q are equivalent if for all w :
 $\hat{\delta}(p, w)$ is an accepting state iff $\hat{\delta}(q, w)$ is an accepting state
- $\hat{\delta}$ is the transition function extended for words

DFA Minimization: Procedure

- Write down all pairs of state as a table
- Every cell in table denotes if corresponding states are equivalent
- Table is initially unmarked
- We mark pair (p_i, p_j) when we discover p_i and p_j are not equivalent

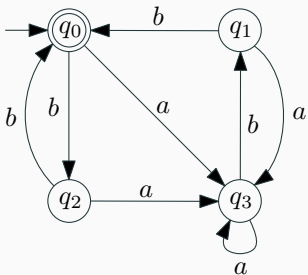
	q_0	q_1	q_2	q_3	q_4	q_5	q_6
q_0	?	?	?	?	?	?	?
q_1		?	?	?	?	?	?
q_2			?	?	?	?	?
q_3				?	?	?	?
q_4					?	?	?
q_5						?	?

DFA Minimization: Procedure

1. Start by marking all cells (q_i, q_j) where one of them is final and other is non-final.
2. Look for unmarked pairs (q_i, q_j) such that for some $c \in \Sigma$, the pair $(\delta(q_i, c), \delta(q_j, c))$ is marked. Then mark (q_i, q_j) .
3. Repeat step 2 until no such unmarked pairs remain.

Illustration of minimization algorithm

First mark accepting/non-accepting pairs



	q_1	q_2	q_3
q_0	✓	✓	✓
q_1			
q_2			

Illustration of minimization algorithm

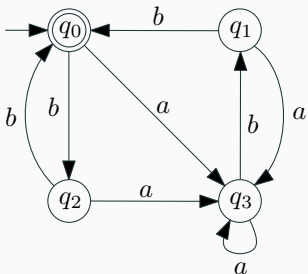
(q_1, q_3) is unmarked,

$q_1 \xrightarrow{b} q_0$,

$q_3 \xrightarrow{b} q_1$,

and (q_0, q_1) is marked,

so mark (q_1, q_3)



q_0	q_1	q_2	q_3
q_0	✓	✓	✓
q_1			
q_2			

Illustration of minimization algorithm

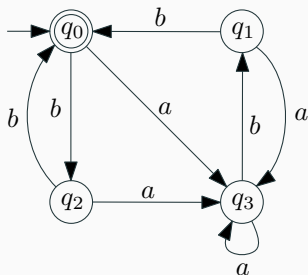
(q_1, q_3) is unmarked,

$q_1 \xrightarrow{b} q_0$,

$q_3 \xrightarrow{b} q_1$,

and (q_0, q_1) is marked,

so mark (q_1, q_3)



q_0	q_1	q_2	q_3
q_0	✓	✓	✓
q_1			✓
q_2			

Illustration of minimization algorithm

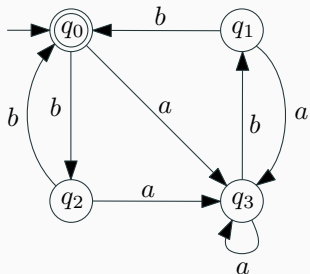
(q_2, q_3) is unmarked,

$q_2 \xrightarrow{b} q_0$,

$q_3 \xrightarrow{b} q_1$,

and (q_0, q_1) is marked,

so mark (q_2, q_3)



q_0	q_1	q_2	q_3
q_0	✓	✓	✓
q_1			✓
q_2			

Illustration of minimization algorithm

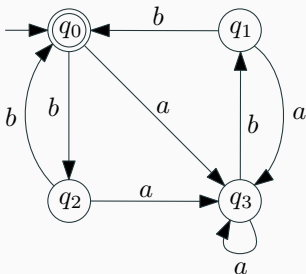
(q_2, q_3) is unmarked,

$q_2 \xrightarrow{b} q_0$,

$q_3 \xrightarrow{b} q_1$,

and (q_0, q_1) is marked,

so mark (q_2, q_3)

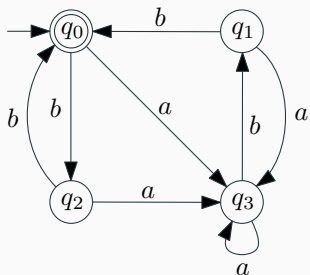


q_0	q_1	q_2	q_3
q_0	✓	✓	✓
q_1			✓
q_2			✓

Illustration of minimization algorithm

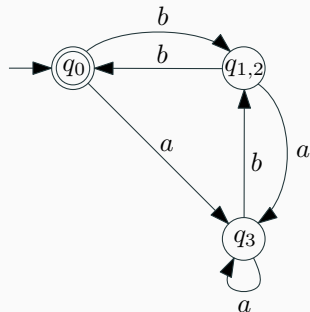
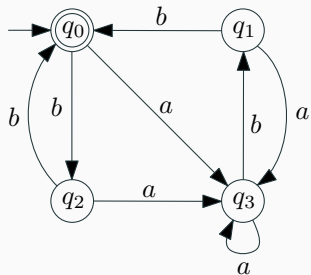
There is no way to mark the only unmarked pair (q_1, q_2)

Obtain minimized DFA by collapsing q_1, q_2 to a single state



	q_1	q_2	q_3
q_0	✓	✓	✓
q_1			✓
q_2			✓

Illustration of minimization algorithm



Exercise

Convert the following DFA to a DFA with 3 states

