Deterministic Finite Automata (DFAs)

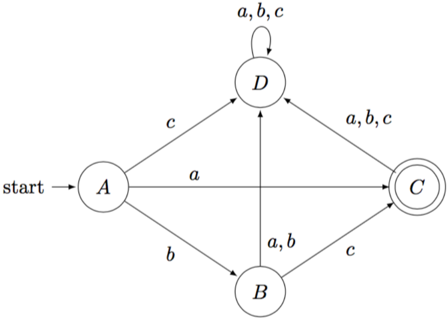
Grammars *generate* strings.
Automata *recognize* strings:
Given some input string x, an automata M either outputs

- "accept" if $x \in \mathcal{L}(M)$, or
- "reject" if $x \notin \mathcal{L}(M)$.

Deterministic Finite Automata (DFAs)

Grammars *generate* strings.
Automata *recognize* strings:
Given some input string x, an automata M either outputs

- "accept" if $x \in \mathcal{L}(M)$, or
- "reject" if $x \notin \mathcal{L}(M)$.



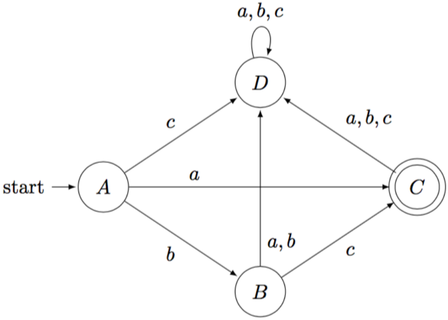
An automata has:

- A set of states.

Deterministic Finite Automata (DFAs)

Grammars *generate* strings.
Automata *recognize* strings:
Given some input string x, an automata M either outputs

- "accept" if $x \in \mathcal{L}(M)$, or
- "reject" if $x \notin \mathcal{L}(M)$.



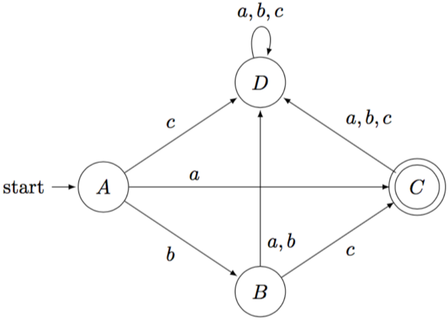
An automata has:

- A set of states.
  - One special state is the start state.
  - A subset of the states are "accept" states. The remainder are "reject" states.

Deterministic Finite Automata (DFAs)

Grammars *generate* strings.
Automata *recognize* strings:
Given some input string x, an automata M either outputs

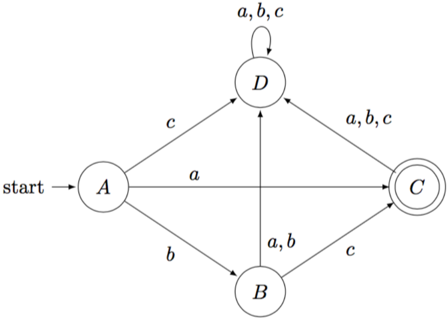
- "accept" if $x \in \mathcal{L}(M)$, or
- "reject" if $x \notin \mathcal{L}(M)$.



An automata has:

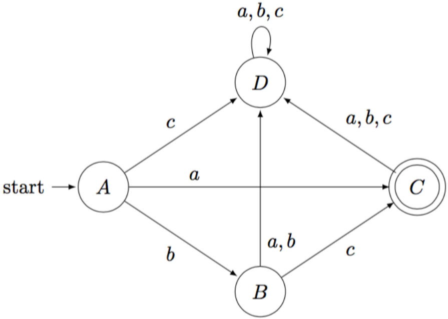
- A set of states.
  - One special state is the start state.
  - A subset of the states are "accept" states. The remainder are "reject" states.
- A set of labeled transitions from one state to another.

DFAs: removing the trap state.



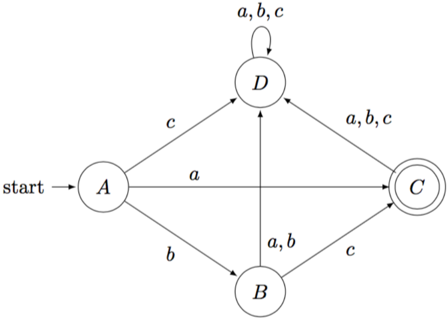
On input x, follow the transitions as you process each character of x, in order.
Accept if and only if you end in an accept state.

DFAs: removing the trap state.



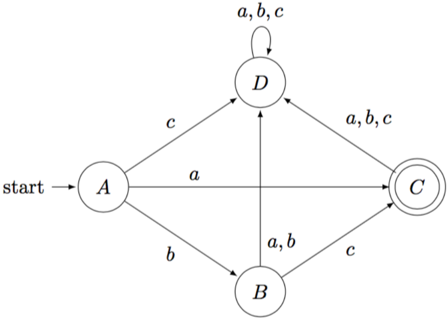
On input x, follow the transitions as you process each character of x, in order.
Accept if and only if you end in an accept state.
$\mathcal{L}(M) = \{a, bc\}$

DFAs: removing the trap state.



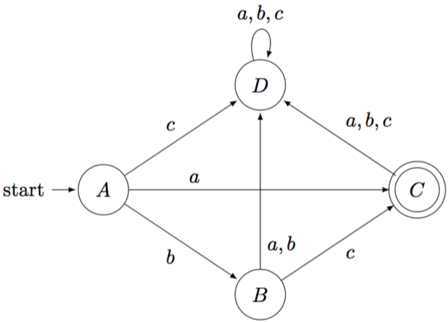
On input x, follow the transitions as you process each character of x, in order.
Accept if and only if you end in an accept state.
$\mathcal{L}(M) = \{a, bc\}$
Note that D is a special state, called a “trap” state.

DFAs: removing the trap state.



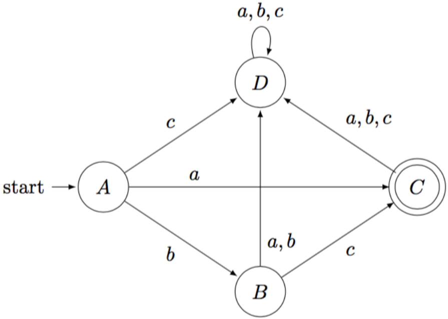
On input x, follow the transitions as you process each character of x, in order.
Accept if and only if you end in an accept state.
$\mathcal{L}(M) = \{a, bc\}$
Note that D is a special state, called a "trap" state.
$c(a + b + c)^*$ terminates in D and is rejected.

DFAs: removing the trap state.



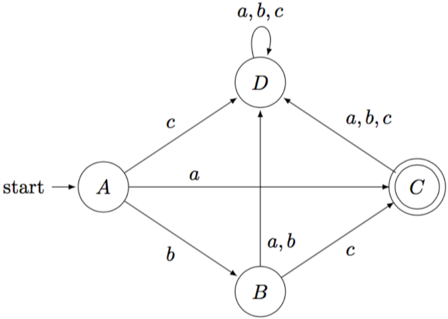
On input x, follow the transitions as you process each character of x, in order.
Accept if and only if you end in an accept state.
$\mathcal{L}(M) = \{a, bc\}$
Note that D is a special state, called a "trap" state.
$c(a + b + c)^*$ terminates in D and is rejected.
$b(a + b)(a + b + c)^*$ terminates in D and is rejected.

DFAs: removing the trap state.



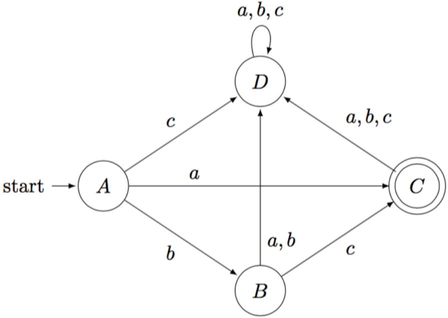
On input x, follow the transitions as you process each character of x, in order.
Accept if and only if you end in an accept state.
$\mathcal{L}(M) = \{a, bc\}$
Note that D is a special state, called a "trap" state.
$c(a + b + c)^*$ terminates in D and is rejected.
$b(a + b)(a + b + c)^*$ terminates in D and is rejected.
$a(a + b + c)(a + b + c)^*$ terminates in D and is rejected.

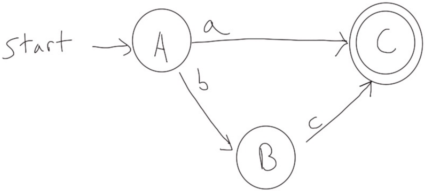
DFAs: removing the trap state.



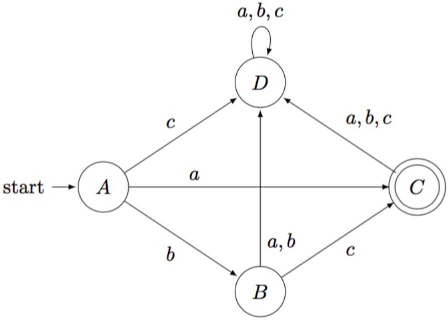
On input x, follow the transitions as you process each character of x, in order.
Accept if and only if you end in an accept state.
$\mathcal{L}(M) = \{a, bc\}$
Note that D is a special state, called a "trap" state.
$c(a+b+c)^*$ terminates in D and is rejected.
$b(a+b)(a+b+c)^*$ terminates in D and is rejected.
$a(a+b+c)(a+b+c)^*$ terminates in D and is rejected.
$bc(a+b+c)(a+b+c)^*$ terminates in D and is rejected.

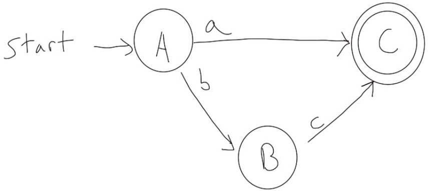
DFAs





DFAs





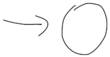
If there is no transition from the current state labeled with the current input character, simply reject.

Example 2

$L = \{x \mid x \in \{a, b\}^* \text{ and every } a \text{ precedes every } b\}$

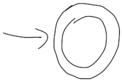
Example 2

$L = \{x \mid x \in \{a, b\}^* \text{ and every } a \text{ precedes every } b\}$



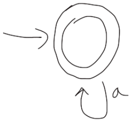
Example 2

$L = \{x \mid x \in \{a, b\}^* \text{ and every } a \text{ precedes every } b\}$



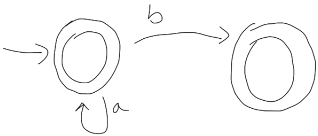
Example 2

$$L = \{x \mid x \in \{a,b\}^* \text{ and every } a \text{ precedes every } b\}$$



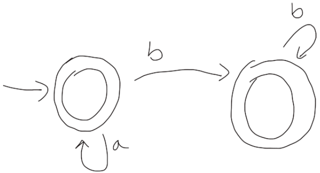
Example 2

$$L = \{x \mid x \in \{a, b\}^* \text{ and every } a \text{ precedes every } b\}$$



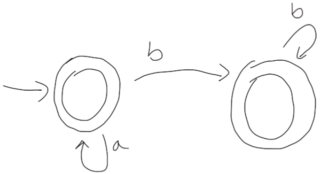
Example 2

$$L = \{x \mid x \in \{a,b\}^* \text{ and every } a \text{ precedes every } b\}$$



Example 2

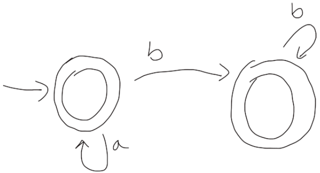
$L = \{x \mid x \in \{a,b\}^* \text{ and every } a \text{ precedes every } b\}$



Alternative ways of specifying the same language:
$L = \{x \mid x = yz \wedge y \in \{a\}^* \wedge z \in \{b\}^*\}$

Example 2

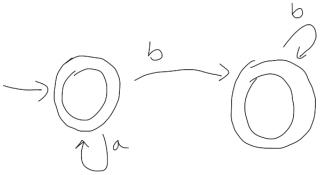
$L = \{x \mid x \in \{a, b\}^* \text{ and every } a \text{ precedes every } b\}$



Alternative ways of specifying the same language:
$L = \{x \mid x = yz \wedge y \in \{a\}^* \wedge z \in \{b\}^*\}$
$L = \{x \mid x \in \mathcal{L}(a^*b^*)\}$

Example 2

$L = \{x \mid x \in \{a,b\}^* \text{ and every } a \text{ precedes every } b\}$



Alternative ways of specifying the same language:
$L = \{x \mid x = yz \wedge y \in \{a\}^* \wedge z \in \{b\}^*\}$
$L = \{x \mid x \in \mathcal{L}(a^*b^*)\}$
$L = \{x \mid x \in \{a,b\}^*$ and there is no occurrence of ba in $x\}$

Example 3

$L = \{x \mid x \in \{a, b\}^*$ and every block of bs has even length$\}$

Example 3

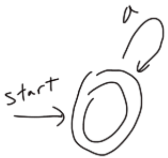
$L = \{x \mid x \in \{a,b\}^*$ and every block of bs has even length$\}$

Example 3

$L = \{x \mid x \in \{a, b\}^* \text{ and every block of } b\text{s has even length}\}$

Example 3

$L = \{x \mid x \in \{a, b\}^* \text{ and every block of } b\text{s has even length}\}$

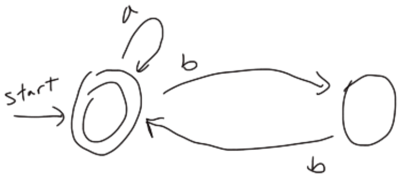


Example 3

$L = \{x \mid x \in \{a, b\}^* \text{ and every block of } b\text{s has even length}\}$

Example 3

$L = \{x \mid x \in \{a, b\}^*$ and every block of bs has even length$\}$



Example 4

L over $\Sigma = \{a, b, c, d\}$ which contains exactly the strings x such that

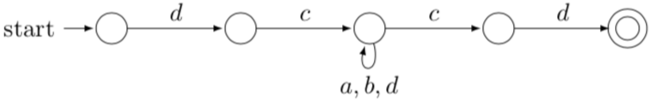
1. x begins with dc
2. x ends in a substring cd (but $x \neq dcd$) and
3. x has no other occurrence of cd between these 2 substrings.

Example 4

L over $\Sigma = \{a, b, c, d\}$ which contains exactly the strings x such that

1. x begins with dc
2. x ends in a substring cd (but $x \neq dcd$) and
3. x has no other occurrence of cd between these 2 substrings.

First attempt (WRONG!):

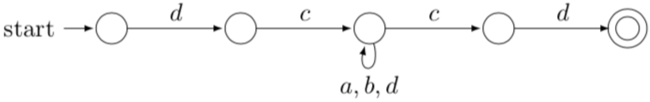


Example 4

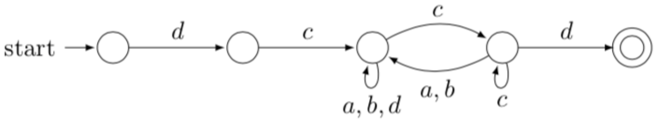
L over $\Sigma = \{a, b, c, d\}$ which contains exactly the strings x such that

1. x begins with dc
2. x ends in a substring cd (but $x \neq dcd$) and
3. x has no other occurrence of cd between these 2 substrings.

First attempt (WRONG!):



A correct machine:

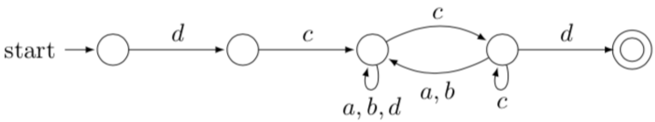


Example 4b

L over $\Sigma = \{a, b, c, d\}$ which contains exactly the strings x such that

1. x begins with dc
2. x ends in a substring cd (but $x \neq dcd$) and
3. x has no other occurrence of cd. (Previous example said: "x has no other occurrence of cd *between these 2 substrings.")*

First attempt (WRONG!):

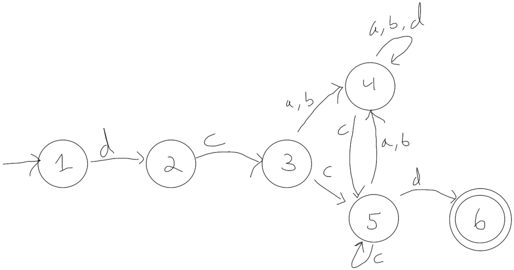


Example 4b

L over $\Sigma = \{a, b, c, d\}$ which contains exactly the strings x such that

1. x begins with dc
2. x ends in a substring cd (but $x \neq dcd$) and
3. x has no other occurrence of cd. (Previous example said: "x has no other occurrence of cd *between these 2 substrings.*")

A correct machine:

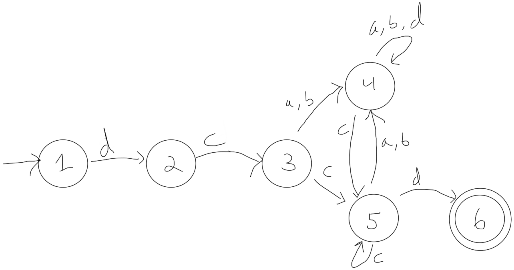


Example 4b

L over $\Sigma = \{a, b, c, d\}$ which contains exactly the strings x such that

1. x begins with dc
2. x ends in a substring cd (but $x \neq dcd$) and
3. x has no other occurrence of cd. (Previous example said: "x has no other occurrence of cd *between these 2 substrings."*)

A correct machine:



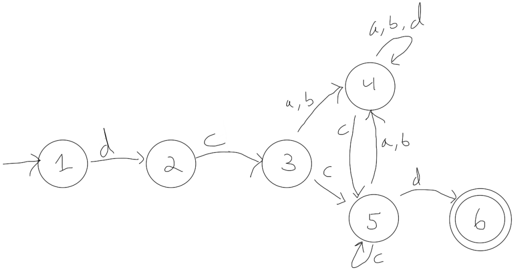
Claim: If $M(x) = 1$, then $x \in L$.

Example 4b

L over $\Sigma = \{a, b, c, d\}$ which contains exactly the strings x such that

1. x begins with dc
2. x ends in a substring cd (but $x \neq dcd$) and
3. x has no other occurrence of cd. (Previous example said: "x has no other occurrence of cd *between these 2 substrings.*")

A correct machine:



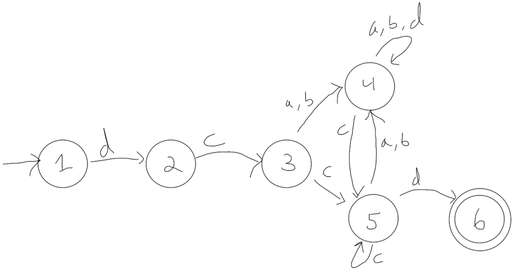
Claim: If $M(x) = 1$, then $x \in L$.
Property 1: Clearly x starts with dc.

Example 4b

L over $\Sigma = \{a, b, c, d\}$ which contains exactly the strings x such that

1. x begins with dc
2. x ends in a substring cd (but $x \neq dcd$) and
3. x has no other occurrence of cd. (Previous example said: "x has no other occurrence of cd *between these 2 substrings.*")

A correct machine:



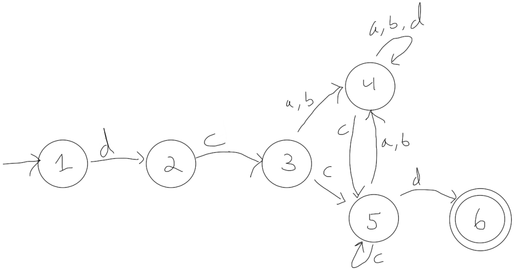
Claim: If $M(x) = 1$, then $x \in L$.
Property 1: Clearly x starts with dc.
Property 2: It must end with cd, because any string leading to state 5 must end in c: all transitions to 5 are labeled c. We can verify by hand that $M(dcd) = 0$.

Example 4b

L over $\Sigma = \{a, b, c, d\}$ which contains exactly the strings x such that

1. x begins with dc
2. x ends in a substring cd (but $x \neq dcd$) and
3. x has no other occurrence of cd. (Previous example said: "x has no other occurrence of cd *between these 2 substrings.*")

A correct machine:



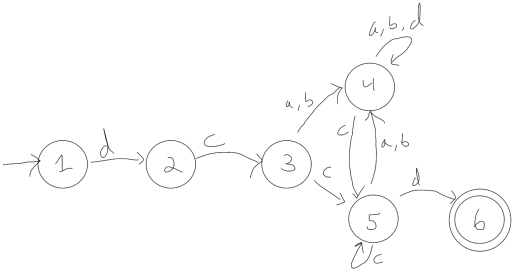
Claim: If $M(x) = 1$, then $x \in L$.
Property 1: Clearly x starts with dc.
Property 2: It must end with cd, because any string leading to state 5 must end in c: all transitions to 5 are labeled c. We can verify by hand that $M(dcd) = 0$.
Property 3: any sub-string ending in c is either in state 3 or 5. Neither of those states have an out-transition labeled d, except the one leading to 6.

Example 4b

L over $\Sigma = \{a, b, c, d\}$ which contains exactly the strings x such that

1. x begins with dc
2. x ends in a substring cd (but $x \neq dcd$) and
3. x has no other occurrence of cd. (Previous example said: "x has no other occurrence of cd *between these 2 substrings.")*

A correct machine:

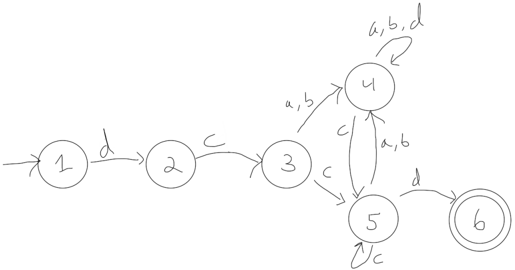


Example 4b

L over $\Sigma = \{a, b, c, d\}$ which contains exactly the strings x such that

1. x begins with dc
2. x ends in a substring cd (but $x \neq dcd$) and
3. x has no other occurrence of cd. (Previous example said: "x has no other occurrence of cd *between these 2 substrings."*)

A correct machine:



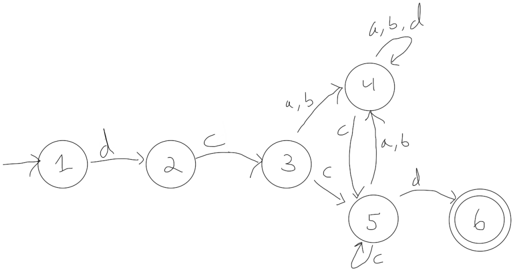
Claim: If $x \in L$, then $M(x) = 1$.

Example 4b

L over $\Sigma = \{a, b, c, d\}$ which contains exactly the strings x such that

1. x begins with dc
2. x ends in a substring cd (but $x \neq dcd$) and
3. x has no other occurrence of cd. (Previous example said: "x has no other occurrence of cd *between these 2 substrings.")*

A correct machine:



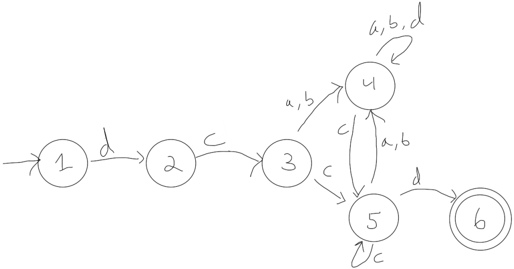
Claim: If $x \in L$, then $M(x) = 1$.
Let w be such that $x = dcwcd$. The first dc leave us in state 3.

Example 4b

L over $\Sigma = \{a, b, c, d\}$ which contains exactly the strings x such that

1. x begins with dc
2. x ends in a substring cd (but $x \neq dcd$) and
3. x has no other occurrence of cd. (Previous example said: "x has no other occurrence of cd *between these 2 substrings.")*

A correct machine:



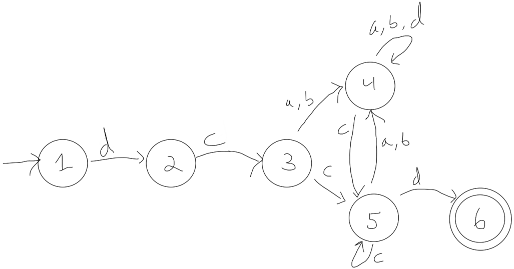
Claim: If $x \in L$, then $M(x) = 1$.
Let w be such that $x = dcwcd$. The first dc leave us in state 3.
By the definition of L, w cannot start with a d, so x is not rejected when M is in state 3. Also, M will never return to state 3.

Example 4b

L over $\Sigma = \{a, b, c, d\}$ which contains exactly the strings x such that

1. x begins with dc
2. x ends in a substring cd (but $x \neq dcd$) and
3. x has no other occurrence of cd. (Previous example said: "x has no other occurrence of cd *between these 2 substrings.")*

A correct machine:



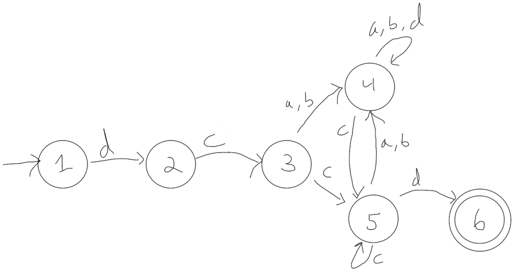
Claim: If $x \in L$, then $M(x) = 1$.
Let w be such that $x = dcwcd$. The first dc leave us in state 3.
By the definition of L, w cannot start with a d, so x is not rejected when M is in state 3. Also, M will never return to state 3.
The other two states in the triangle have 4 transitions out, so neither causes a rejection on w.

Example 4b

L over $\Sigma = \{a, b, c, d\}$ which contains exactly the strings x such that

1. x begins with dc
2. x ends in a substring cd (but $x \neq dcd$) and
3. x has no other occurrence of cd. (Previous example said: "x has no other occurrence of cd *between these 2 substrings."*)

A correct machine:



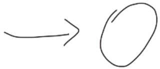
Claim: If $x \in L$, then $M(x) = 1$.
Let w be such that $x = dcwcd$. The first dc leave us in state 3.
By the definition of L, w cannot start with a d, so x is not rejected when M is in state 3. Also, M will never return to state 3.
The other two states in the triangle have 4 transitions out, so neither causes a rejection on w.
Finally, since we know that x ends in cd, regardless of where in the triangle w leaves us (even if $w = \Lambda$), cd carries us to the accept state.

Example 5

$$(ab)^* + c^*$$

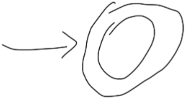
Example 5

$$(ab)^* + c^*$$



Example 5

$$(ab)^* + c^*$$

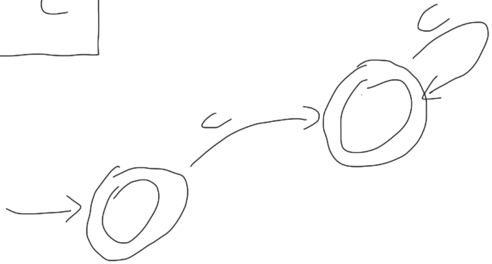


Example 5

$$\boxed{(ab)^* + c^*}$$

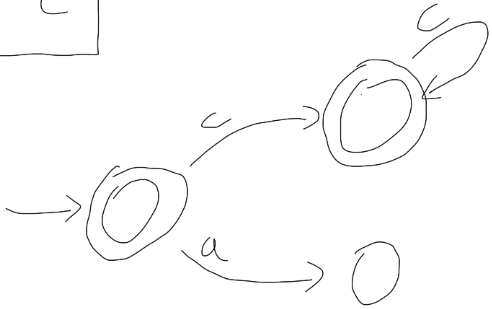
Example 5

$\boxed{(ab)^* + c^*}$



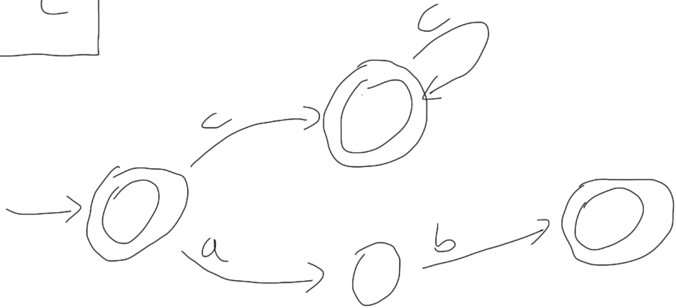
Example 5

$\boxed{(ab)^* + c^*}$



Example 5

$$\boxed{(ab)^* + c^*}$$



Example 5

$(ab)^* + c^*$