Closure Properties of DFAs

Sipser pages 44 - 47

Closure properties of DFAs

Languages captured by DFA's are closed under

- Union
- Concatenation
- Kleene Star
- Complement
- Intersection

That is to say if L_1 and L_2 are recognized by a DFA, then there exists another DFA, L_3, such that

1. L_3 = complement L_1 $\{ x \mid x \notin L_1 \}$
2. $L_3 = L_1 \cup L_2$ $\{ x \mid x \in L_1$ or $x \in L_2 \}$
3. $L_3 = L_1 \cap L_2$ $\{ x \mid x \in L_1$ and $x \in L_2 \}$
4. $L_3 = L_1^*$ (The first 3 are easy, we'll wait on 4 and 5)
5. $L_3 = L_1 \bullet L_2$

Proof Strategy

To prove these properties

1. We'll assume some language (or languages) are recognized by DFAs
2. The then that DFA must be a 5-tuple $\mathbf{A = (Q,\Sigma,\delta,q_0,F)}$
3. Then we'll use the pieces of the 5-tuple to create a new 5-tuple that is the DFA we want.
4. It is very similar to writing a program!

Complement

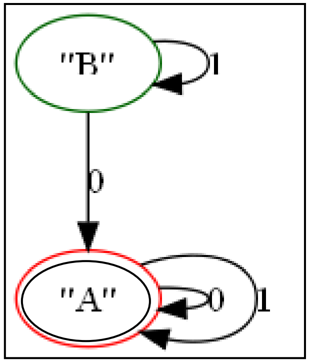
Complementation

Take a DFA for L and change the status - final or non-final - of all its states. The resulting DFA will accept exactly those strings that the first one rejects. It is, therefore, a DFA for the Complent(L).

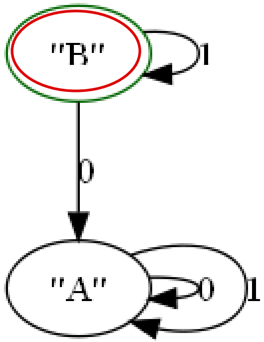
Thus, the complement of DFA recognizable language is DFA recognizable.

Complement Example

Contains a “0”

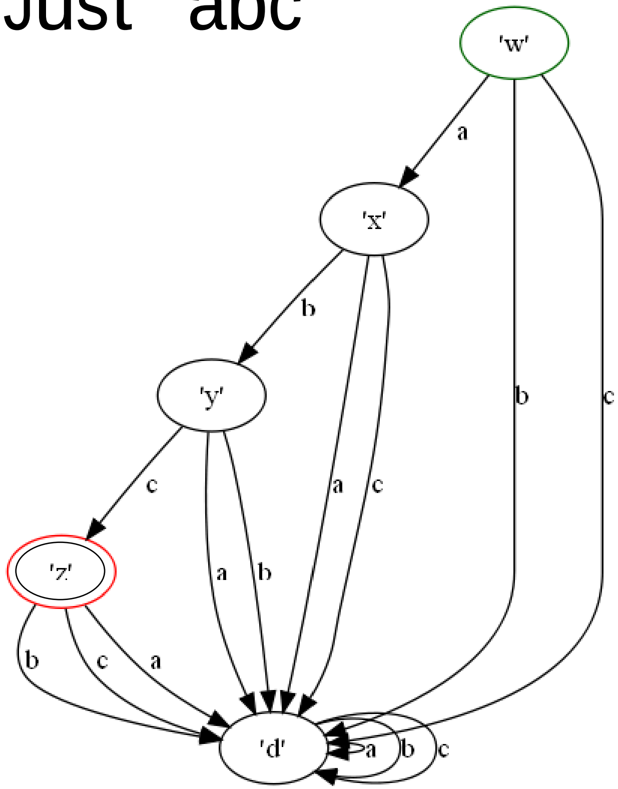


Contains only “1”

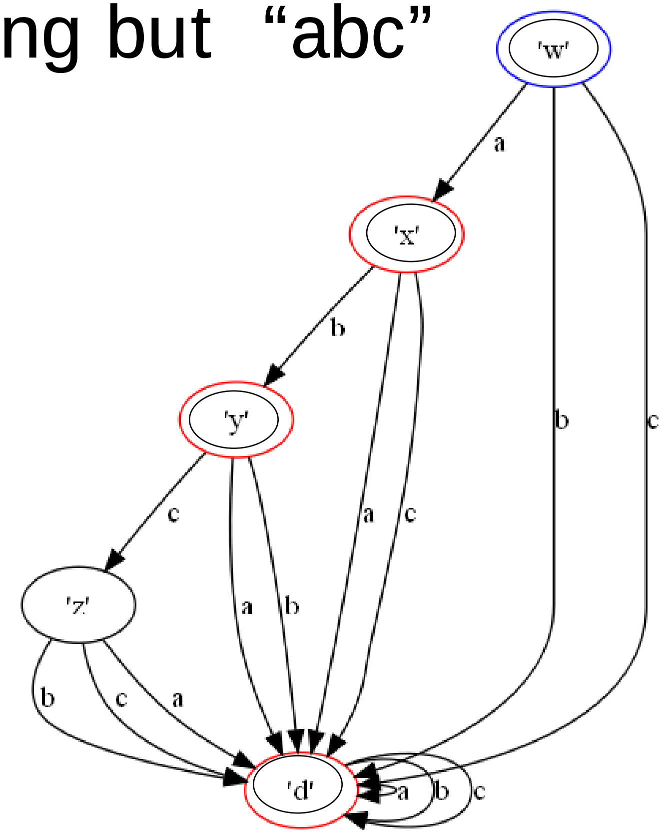


2nd Complement Example

Just "abc"



Anything but "abc"



As a Haskell Program

```
compDFA :: (Ord q) => DFA q s -> DFA q s
compDFA m = DFA (states m)
                (symbols m)
                (trans m)
                (start m)
                new
   where new = [ s
               | s <- states m
                , not(elem s (accept m))]
```

Intersection

The intersection $L \cap M$ of two DFA recognizable languages must be recognizable by a DFA too. A constructive way to show this is to construct a new DFA from 2 old ones.

Constructive Proof

The proof is based on a construction that given two DFAs A and B, produces a third DFA C such that L(C) = L(A) $\cap$ L(B). The states of C are pairs (p,q) , where p is a state of A and q is a state of B. A transition labeled a leads from (p,q) to (p',q') iff there are transitions

$$p \xrightarrow{a} p' \qquad q \xrightarrow{a} q'$$

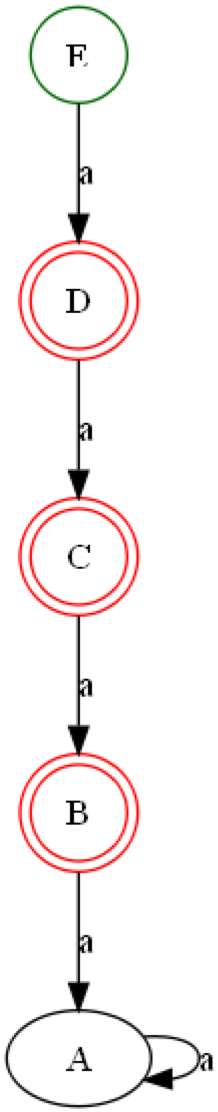
in A and B. The start state is the pair of original start states; the final states are pairs of original final states. The transition function

$$\delta_{A \cap B}(q,a) = (\delta_A(q,a), \delta_B(q,a))$$

This is called the *product construction*.

Example 1

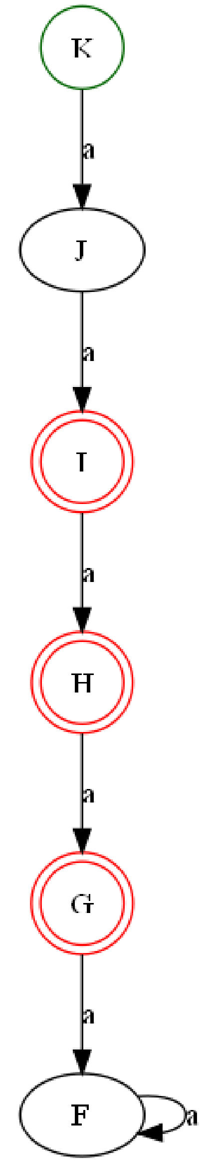
a+aa+aaa

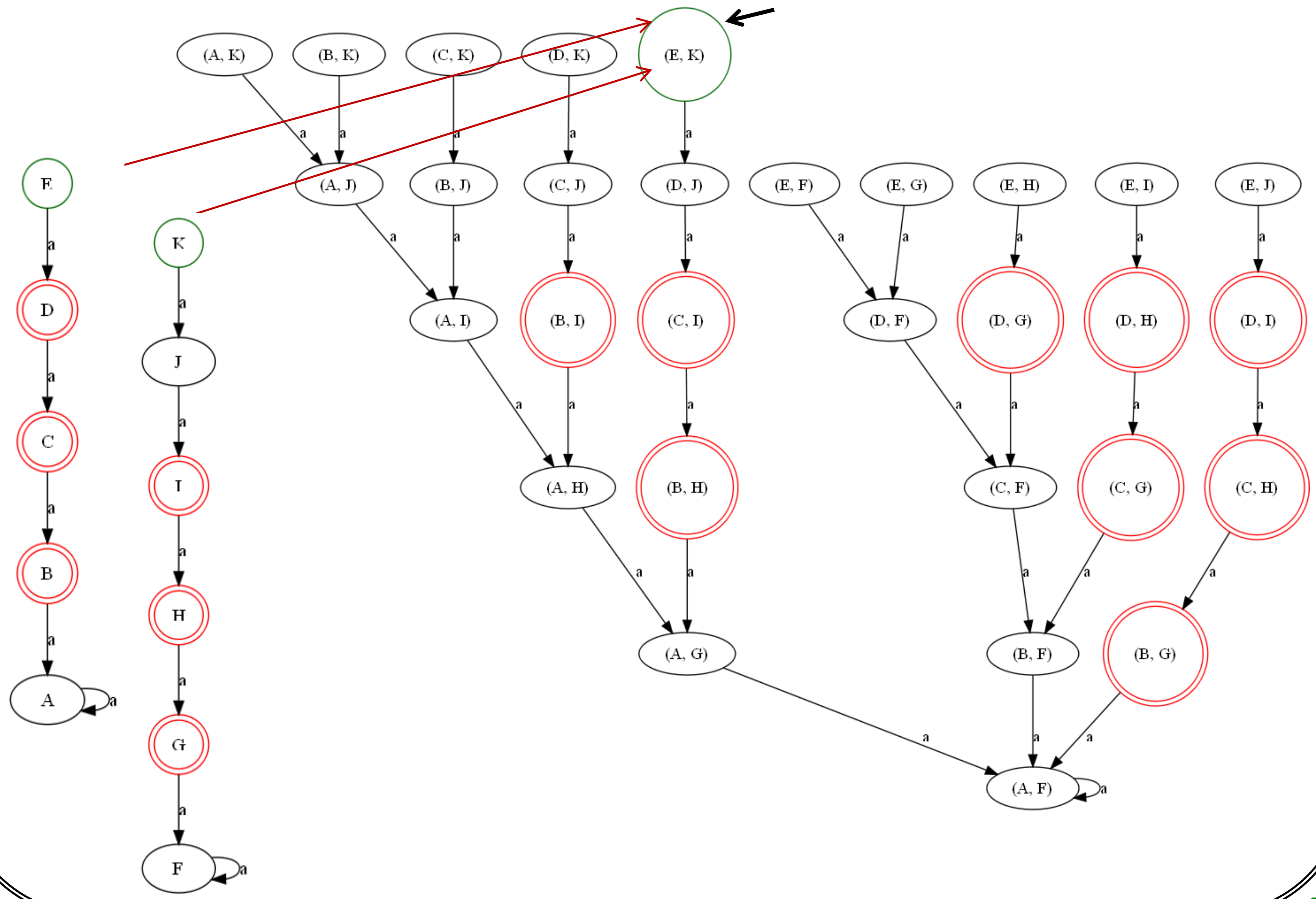


What is the intersection?

Make a new DFA where states of the new DFA are pairs of states form the old ones

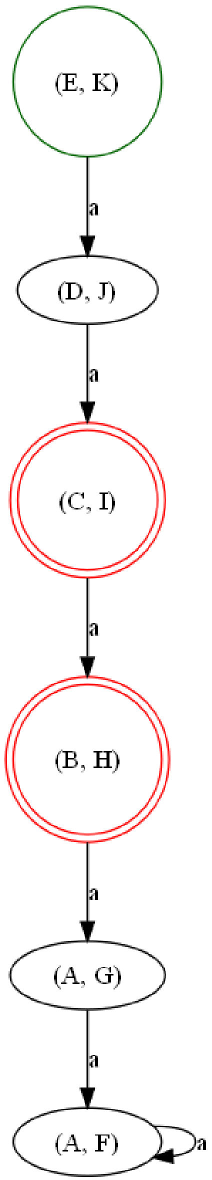
aa+aaa+aaaa





E
D
C
B
A
K
J
I
H
G
F
(A, K)
(B, K)
(C, K)
(D, K)
(E, K)
(A, J)
(B, J)
(C, J)
(D, J)
(E, F)
(E, G)
(E, H)
(E, I)
(E, J)
(A, I)
(B, I)
(C, I)
(D, F)
(D, G)
(D, H)
(D, I)
(A, H)
(B, H)
(C, F)
(C, G)
(C, H)
(A, G)
(B, F)
(B, G)
(A, F)
a

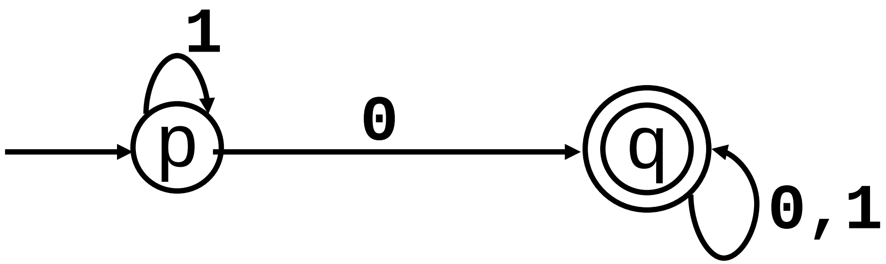
Reachable states only



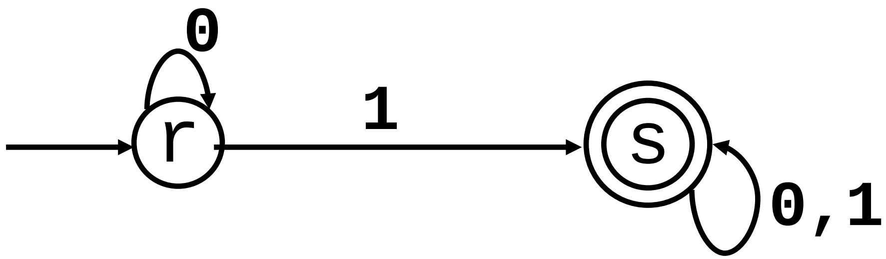
Intersection

{a,aa,aaa} ∩ {aa,aaa,aaaa}

Example 2



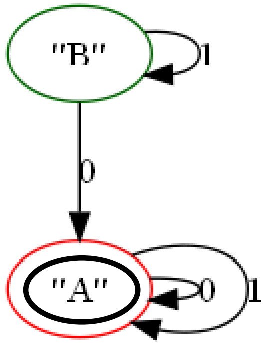
A – string contains a 0



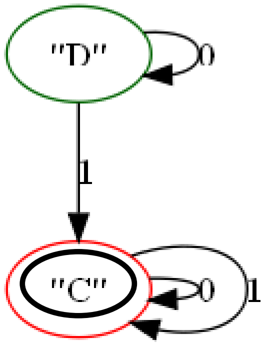
B – string contains a 1

C – string contains a 0 and a 1

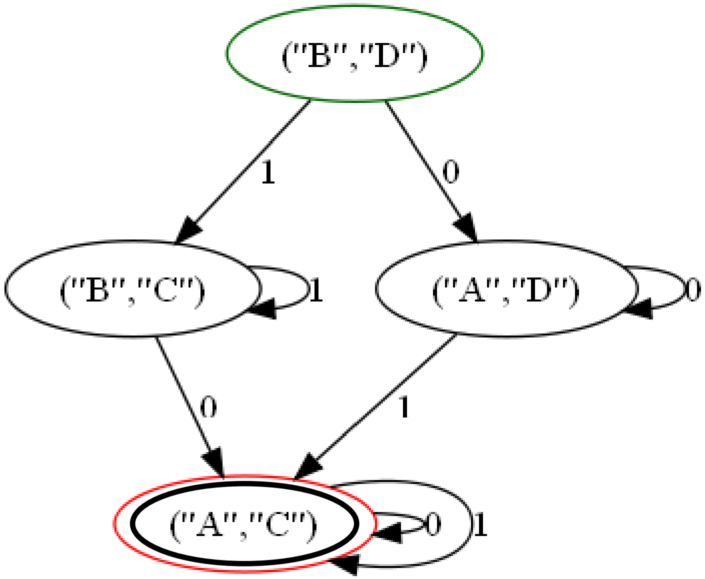
Contains a "0"



Contains a "1"



Contains both a "1" and a "0"



As a Haskell Program

```
intersectDFA (DFA bigQ1 s1 d1 q10 f1)
             (DFA bigQ2 s2 d2 q20 f2)
  =  DFA [(q1,q2) | q1 <- bigQ1, q2 <- bigQ2]
         s1
         (\ (q1,q2) a -> (d1 q1 a, d2 q2 a))
         (q10,q20)
         [(q1,q2) | q1 <- f1, q2 <- f2])
```

Difference

The identity:

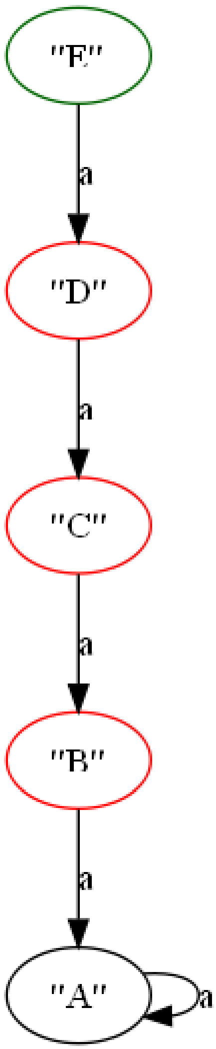
$$L - M = L \cap C(M)$$

reduces the closure under set-theoretical difference operator to closure under complementation and intersection.

Example Difference

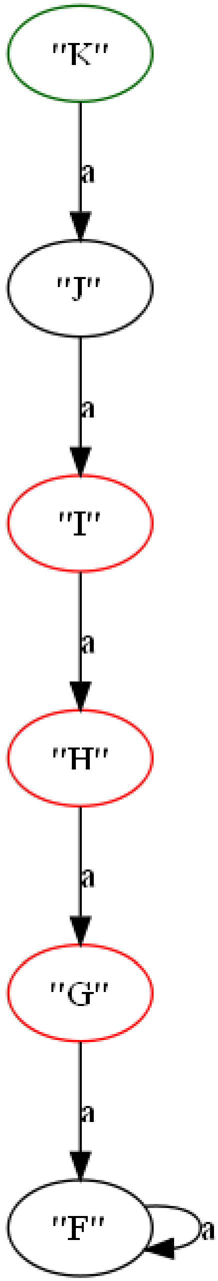
$L - M = L \cap C(M)$

L={a,aa,aaa}

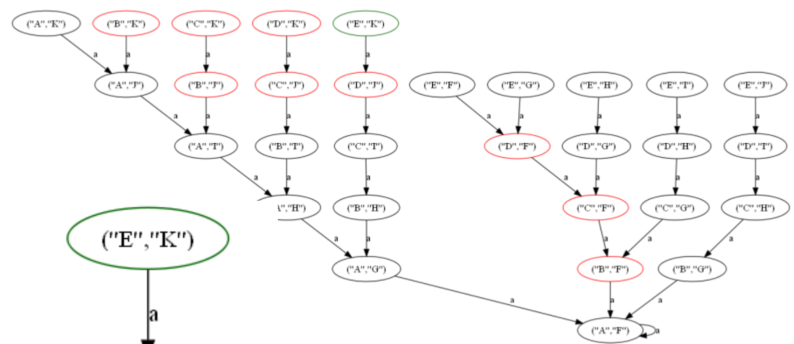


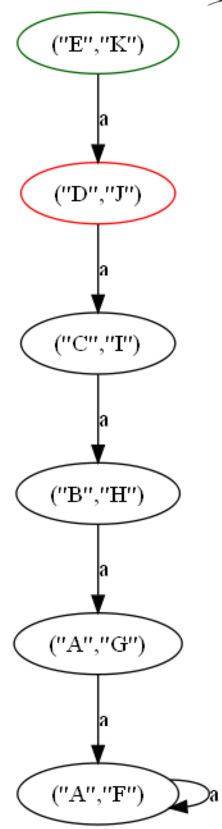
-

M={aa,aaa,aaaa}



=





Union

- The union of the languages of two DFAs (over the same alphabet) is recognizable by another DFA.
- We reuse the product construction of the intersection proof, but widen what is in the final states of the constructed result.

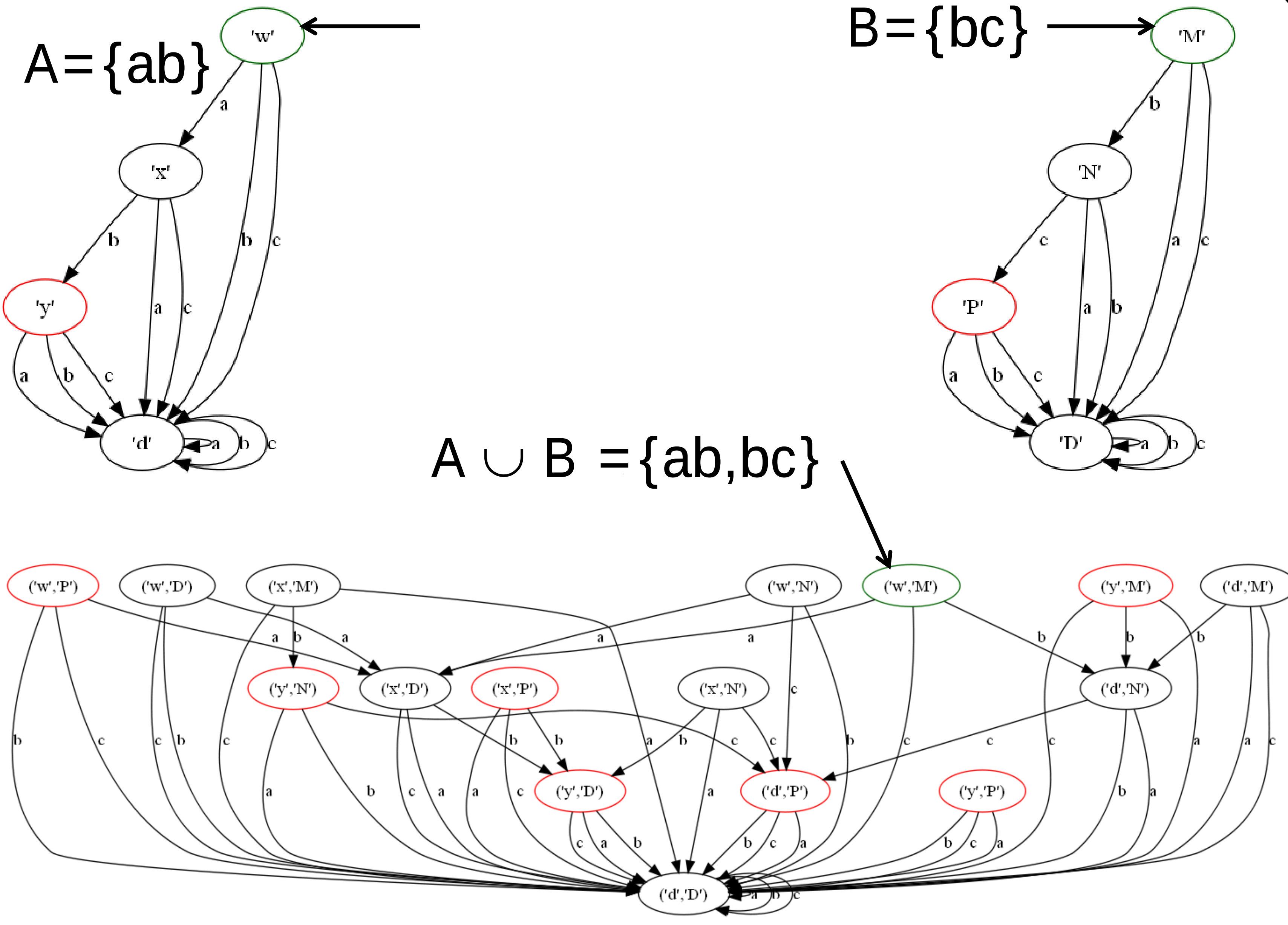
Let $A=(\mathbf{Q_a}, \Sigma, T_a, \mathbf{s_a}, \mathbf{F_a})$ and

$B = (\mathbf{Q_b}, \Sigma, T_b, \mathbf{s_b}, \mathbf{F_b})$

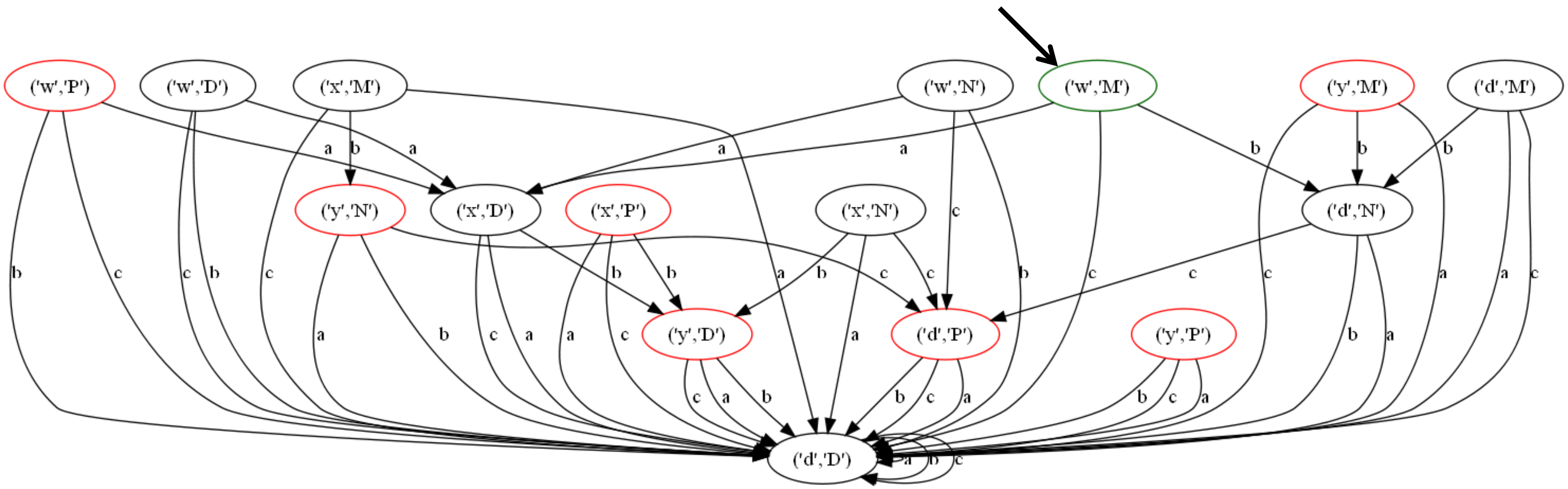
Then: $A \cup B = ((\mathbf{Q_a} \times \mathbf{Q_b}), \Sigma, \delta, (\mathbf{s_a}, \mathbf{s_b}), \mathbf{Final})$

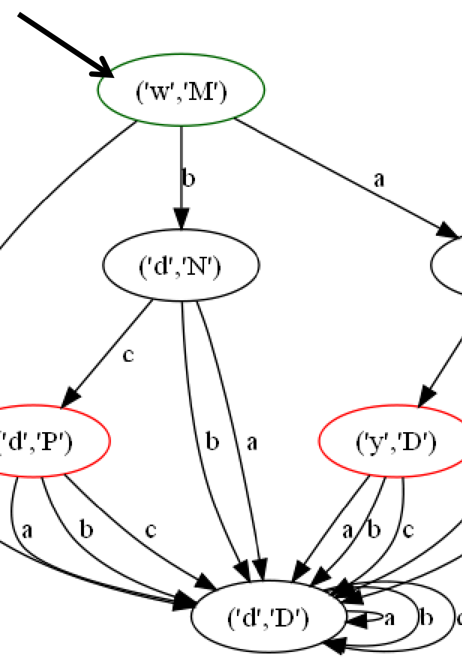
$$\text{Final} = \{ (p,q) \mid p \in F_a,\ q \in Q_b \} \cup \{ (p,q) \mid p \in Q_a,\ q \in F_b \}$$

$$\delta((a,b),x) = (T_a(a,x), T_b(b,y))$$


A={ab}
'w'
'x'
'y'
'd'
B={bc}
'M'
'N'
'P'
'D'
A ∪ B ={ab,bc}
('w','P')
('w','D')
('x','M')
('w','N')
('w','M')
('y','M')
('d','M')
('y','N')
('x','D')
('x','P')
('x','N')
('d','N')
('y','D')
('d','P')
('y','P')
('d','D')

Only reachable from start





As a Haskell Program

```
unionDFA (DFA bigQ1 s1 d1 q10 f1)
         (DFA bigQ2 s2 d2 q20 f2)
  = DFA [(q1,q2) | q1 <- bigQ1, q2 <- bigQ2]
        s1
        (\ (q1,q2) a -> (d1 q1 a, d2 q2 a))
        (q10,q20)
        ([(q1,q2) | q1 <- f1, q2 <- bigQ2] ++
         [(q1,q2) | q1 <- bigQ1, q2 <- f2])
```

Example Closure Construction

Given a language L, let L' be the set of all prefixes of even length strings which belong to L. We prove that if L is regular then L' is also regular.

It is easy to show that prefix(L) is regular when L is (How?). We also know that the language **Even** of even length strings is regular (How?). All we need now is to note that

L' = **Even** $\cap$ prefix(L)

and use closure under intersection.

What's next

We have given constructions for showing that DFAs are closed under

1. Complement
2. Intersection
3. Difference
4. Union

In order to establish the closure properties of

1. Reversal
2. Kleene star
3. Concatenation

We will need to introduce a new computational system.