Computation theory

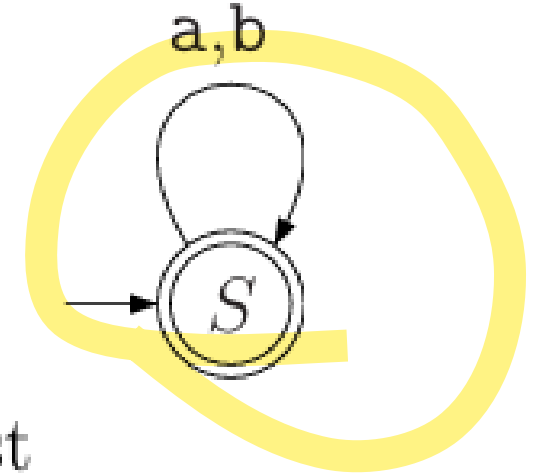
Lecture 3

Brwa R. Hassan

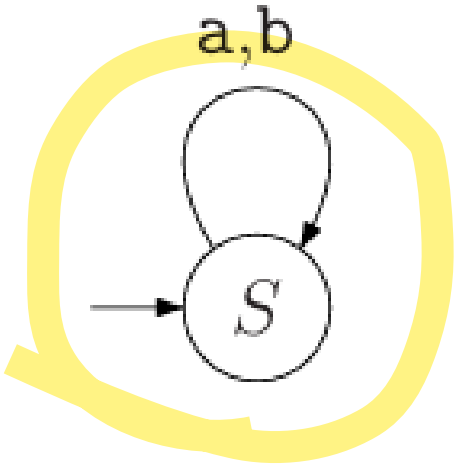
More on DFAs

Some special DFAs

For $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, consider the following DFA that accepts Σ^*:



The DFA that accepts nothing, is just



Formal definition of a DFA

Consider the following automata, that we saw in the previous lecture:

We saw last class that the following components are needed to specify a DFA:

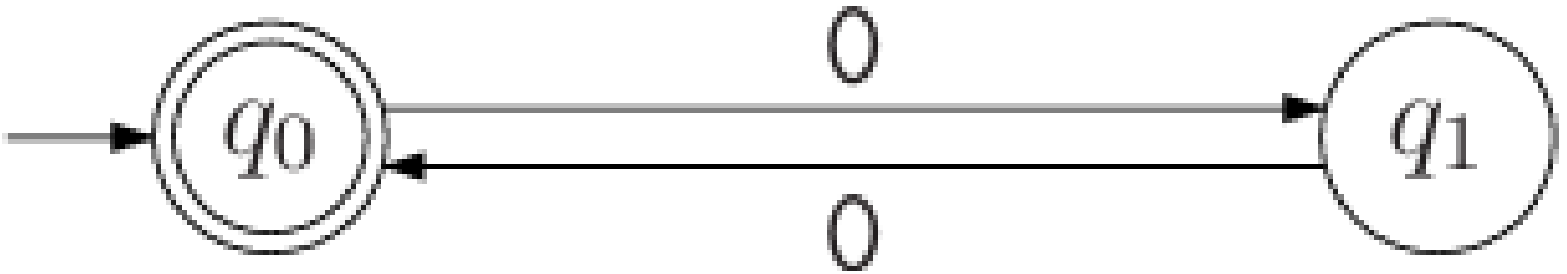
(i) a (finite) alphabet

(ii) a (finite) set of states

(iii) which state is the start state?

(iv) which states are the final states?

(v) what is the transition from each state, on each input character?



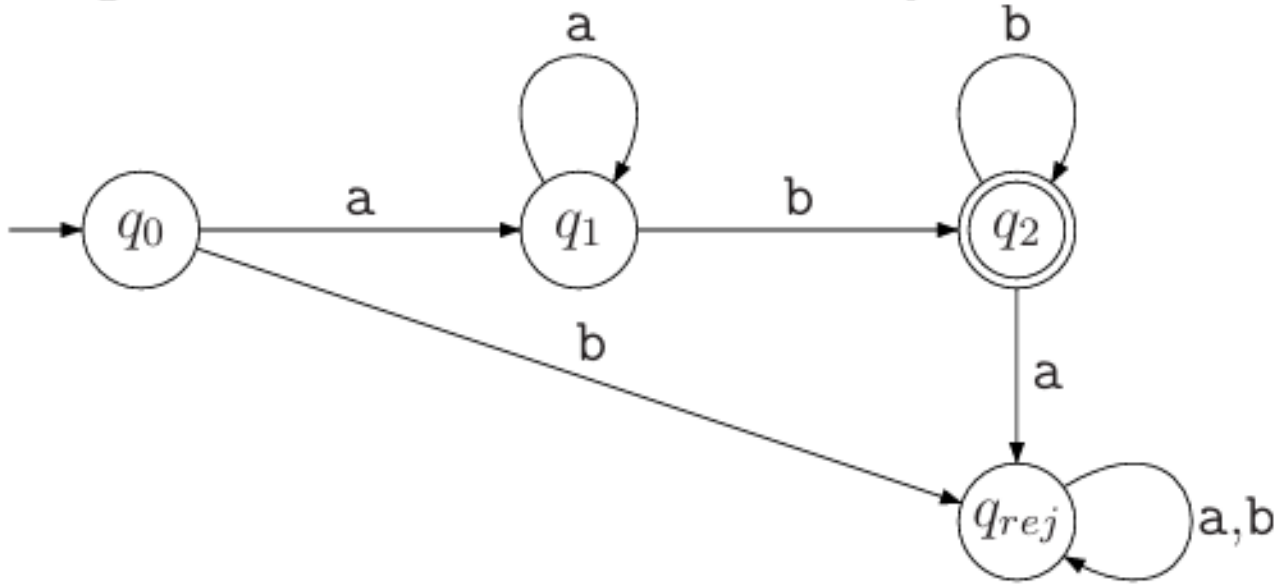
Formal definition of DFA

Formally, a ***deterministic finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q: A finite set (the set of ***states***).
- Σ: A finite set (the ***alphabet***)
- $\delta : Q \times \Sigma \to Q$ is the ***transition function***.
- q_0: The ***start*** state (belongs to Q).
- F: The set of ***accepting*** (or ***final***) states, where $F \subseteq Q$.

Example

For example, let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and consider the following DFA M, whose language $L(M)$ contains strings consisting of one or more a's followed by one or more b's.



Then $M = (Q, \Sigma, \delta, q_0, F)$, $Q = \{q_0, q_1, q_2, q_{rej}\}$, and $F = \{q_2\}$. The transition function δ is defined by

Then $M = (Q, \Sigma, \delta, q_0, F)$, $Q = \{q_0, q_1, q_2, q_{rej}\}$, and $F = \{q_2\}$. The transition function δ is defined by

δ	a	b
q_0	q_1	q_{rej}
q_1	q_1	q_2
q_2	q_{rej}	q_2
q_{rej}	q_{rej}	q_{rej}

We can also define δ using a formula

$$\delta(q_0, a) = q_1$$

$$\delta(q_1, a) = q_1$$

$$\delta(q_1, b) = q_2$$

$$\delta(q_2, b) = q_2$$

$\delta(q, t) = q_{rej}$ for all other values of q and t.

Formal definition of acceptance

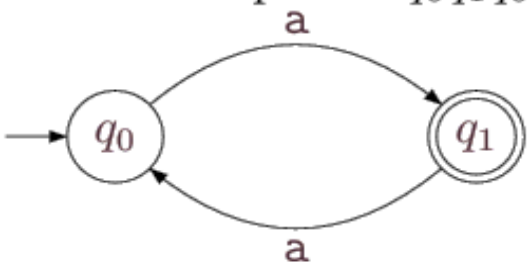
We've also seen informally how to run a DFA. Let us turn that into a formal definition. Suppose $M = (Q, \Sigma, \delta, q_0, F)$ is a given DFA and $w = w_1 w_2 \dots w_k \in \Sigma^*$ is the input string. Then M ***accepts*** w iff there exists a sequence of states $r_0, r_1, \dots r_k$ in Q, such that

1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for $i = 0, \dots, k-1$.
3. $r_k \in F$.

The ***language recognized*** by M, denoted by $L(M)$, is the set $\{w \mid M \text{ accepts } w\}$.

For example, when our automaton above accepts the string aabb, it uses the state sequence $q_0q_1q_1q_2q_2$. (Draw a picture of the transitions.) That is $r_0 = q_0$, $r_1 = q_1$, $r_2 = q_1$, $r_3 = q_2$, and $r_4 = q_2$.

Note that the states do not have to occur in numerical order in this sequence, e.g. the following DFA accepts aaa using the state sequence $q_0q_1q_0q_1$.



A language (i.e. set of strings) is ***regular*** if it is recognized by some DFA.

Closure properties

Closure properties refer to the characteristics of a set of objects (often languages, functions, or mathematical structures) with respect to certain operations. If performing an operation on elements of the set always results in another element that is still within the same set, the set is said to be closed under that operation.

Consider the set of odd integers. If we multiply two odd integers, the answer is always odd. So, the set of odd integers is said to be closed under multiplication. But it is not closed under addition. For example, 3 + 5 = 8 which is not odd.

Closure Under Complement of Regular Languages

- Statement:
  The class of regular languages is closed under complementation.
  That is, if *L* is a regular language over an alphabet Σ , then its complement

$$\overline{L} = \Sigma^* \setminus L = \{w \in \Sigma^* \mid w \notin L\}$$

- **Construction:**

Let *L* be a regular language, and let

M=(*Q*,Σ,*δ*,*q*0,*F*)
be a DFA such that *L*(*M*)=*L* .

Define a new DFA

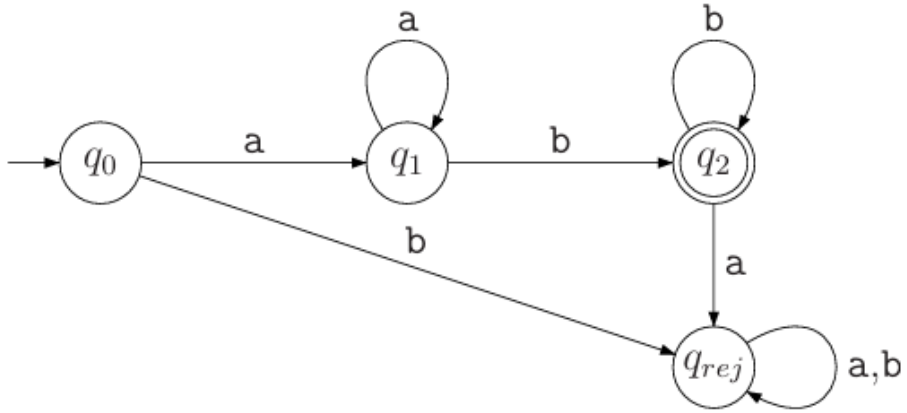
M′=(*Q*,Σ,*δ*,*q*0,*Q**F*)

Example

Let $\Sigma=\{a,b\}$, and let
$L=\{w|w$ contains an even number of a's$\}$.
This is regular (recognized by a 2-state DFA).

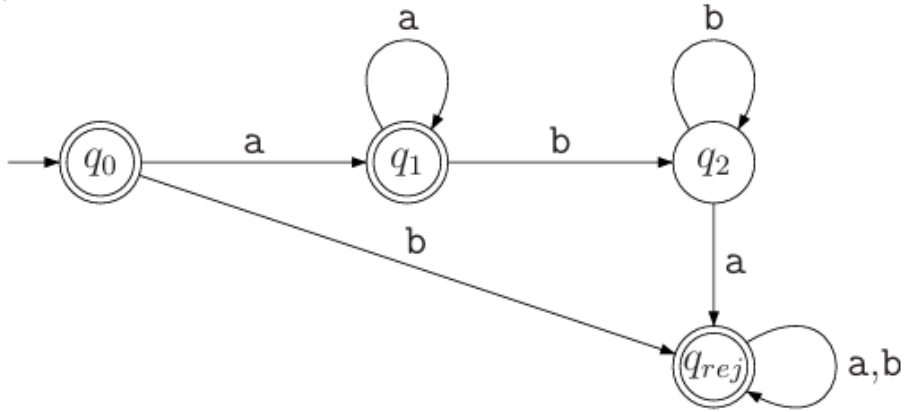
The complement $L=\{w|w$ contains an odd number of a's$\}$
is obtained by swapping the accepting and rejecting states — and is also regular.

Example



The complement language $\overline{L}$ contains the empty string, strings in which some b's precede some a's, and strings that contain only a's or only b's.

Our new DFA M' should accept exactly those strings that M rejects. So we can make M' by swapping final/non-final markings on the states:



Formally, $M' = (Q, \Sigma, \delta, q_0, Q \setminus F)$.

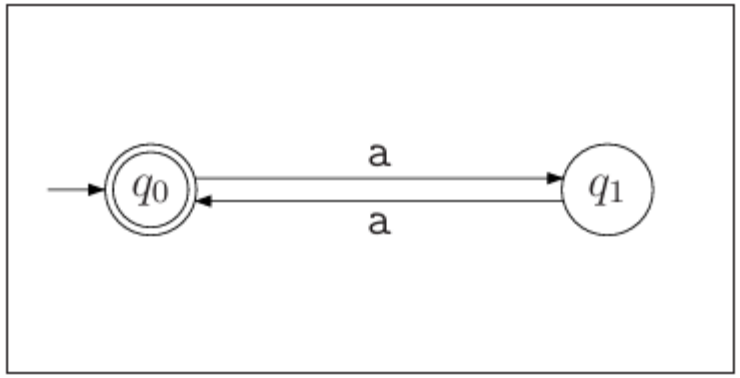
Closure Under Intersection

- Definition:
  A class of languages is closed under intersection if, for any two languages $L1$ and $L2$ in the class, their intersection

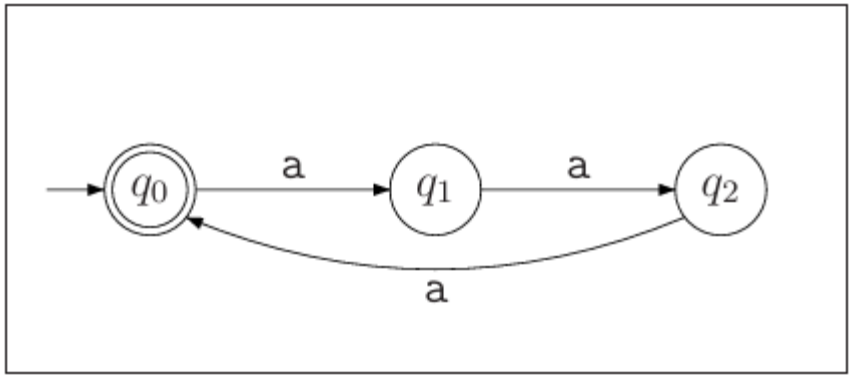
- $L1 \cap L2 = \{w | w \in L1 \text{ and } w \in L2\}$

Example

M_1:



M_2:

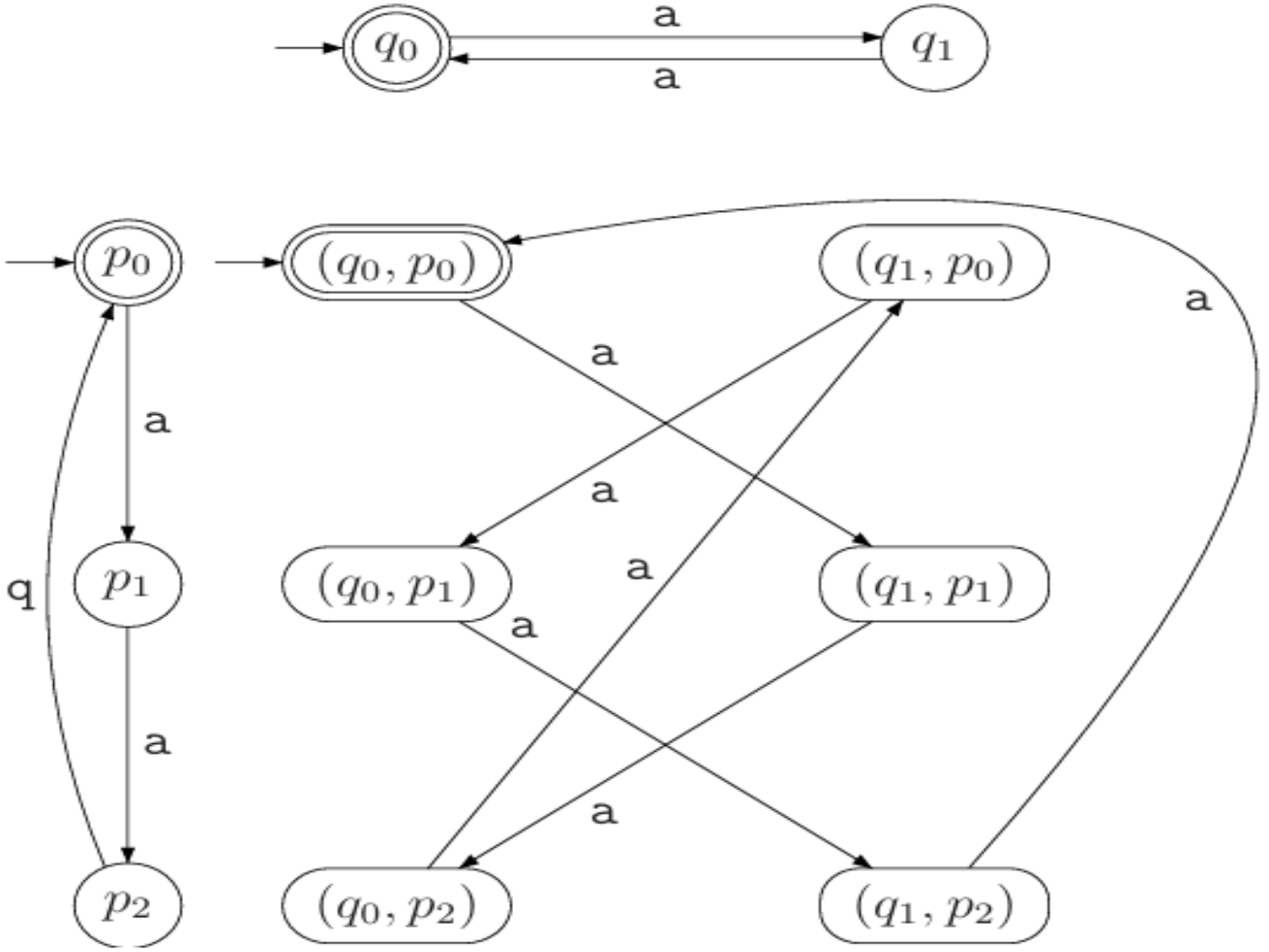


Closure Under Intersection

We combine the two DFAs using Cartesian product of states.

- ✓ **New DFA** $M=(Q,\Sigma,\delta,q0,F)$
- ✓ **States:** $Q=Q1\times Q2$
- ✓ **Start state:** q0,p0
- ✓ **Accepting states:** F=F1×F2 if both final
- ✓ **Transition function:**

$\delta((p,s),x)=(\delta 1(p,x),\delta 2(s,x))$

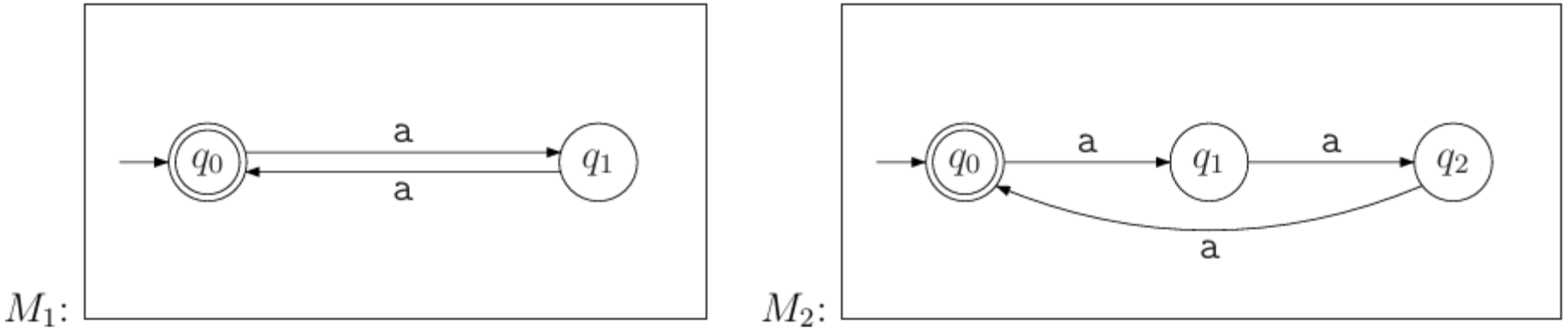


Closure Under Union

- Definition:
  A class of languages is closed under union if, for any two languages $L1$ and $L2$ in the class, their union

- $L1 \cup L2 = \{w | w \in L1 \text{ or } w \in L2\}$

Example

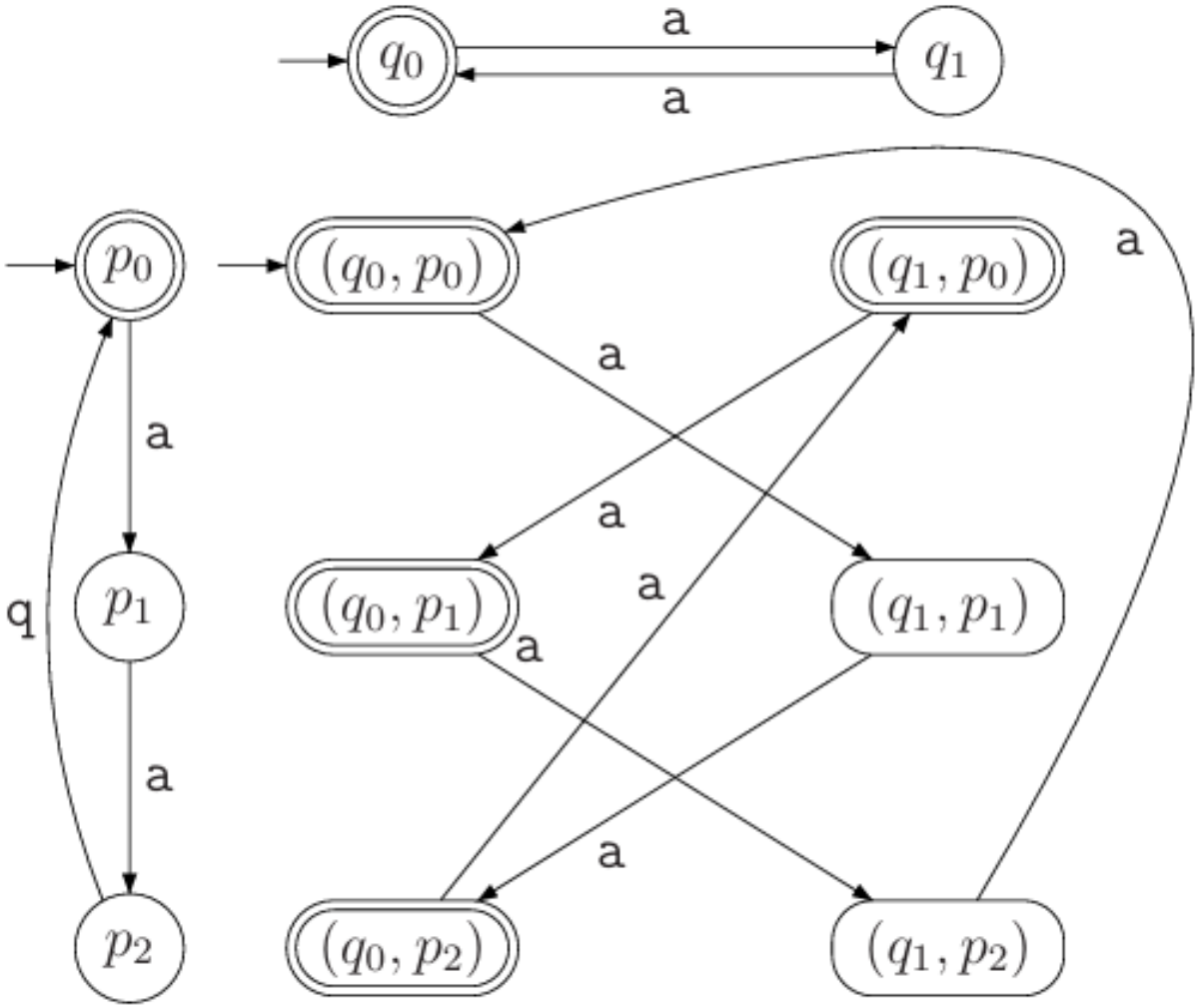


Closure Under Intersection

We combine the two DFAs using Cartesian product of states.

- ✓ **New DFA** $M=(Q,\Sigma,\delta,q0,F)$
- ✓ **States:** $Q=Q1\times Q2$
- ✓ **Start state:** q0,p0
- ✓ **Accepting states:** F=F1×F2 if one final
- ✓ **Transition function:**

$\delta((p,s),x)=(\delta 1(p,x),\delta 2(s,x))$



Thank you