Accepting the Union, Intersection, or Difference of Two Languages

- Suppose that L_1 and L_2 are languages over Σ
  - Given an FA that accepts L_1 and another FA that accepts L_2, we can construct one that accepts $L_1 \cup L_2$, $L_1 \cap L_2$, L_1 - L_2

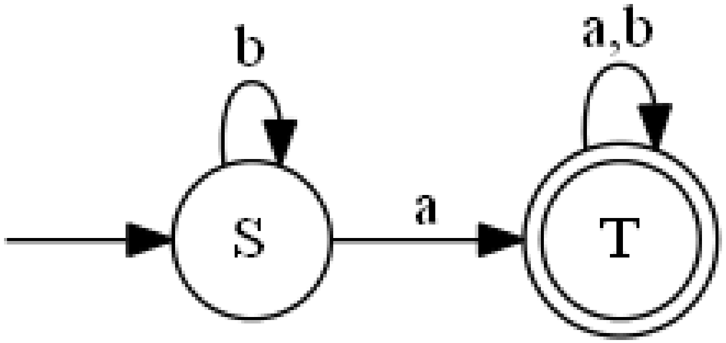
  How to construct an FA for $L_1 \cup L_2$?

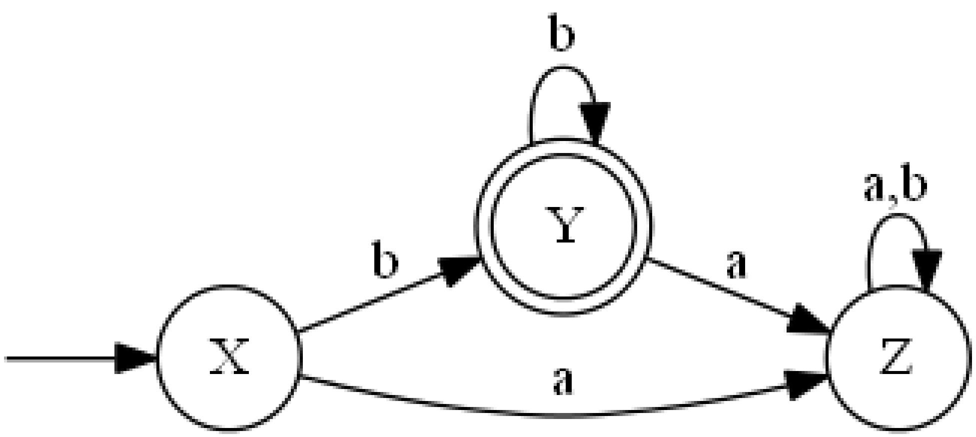
  - The idea is to construct an FA that executes both of the original FAs at the same time
  - This works because if $x \in \Sigma^*$, then knowing whether $x \in L_1$ and whether $x \in L_2$ is enough to determine whether $x \in L_1 \cup L_2$

Accepting the Union, Intersection, or Difference of Two Languages (cont'd.)

- Theorem: Suppose $M_1=(Q_1, \Sigma, q_1, A_1, \delta_1)$ and $M_2=(Q_2, \Sigma, q_2, A_2, \delta_2)$ are FAs accepting L_1 and L_2. Let $M=(Q, \Sigma, q_0, A, \delta)$ be defined as follows:
  - $Q = Q_1 \times Q_2$
  - $q_0 = (q_1, q_2)$
  - $\delta((p, q), \sigma) = (\delta_1(p, \sigma), \delta_2(q, \sigma))$
- Then, if :
  - $A = \{(p, q) \mid p \in A_1 \text{ or } q \in A_2\}$, M accepts $L_1 \cup L_2$
  - $A = \{(p, q) \mid p \in A_1 \text{ and } q \in A_2\}$, M accepts $L_1 \cap L_2$
  - $A = \{(p, q) \mid p \in A_1 \text{ and } q \notin A_2\}$, M accepts $L_1 - L_2$

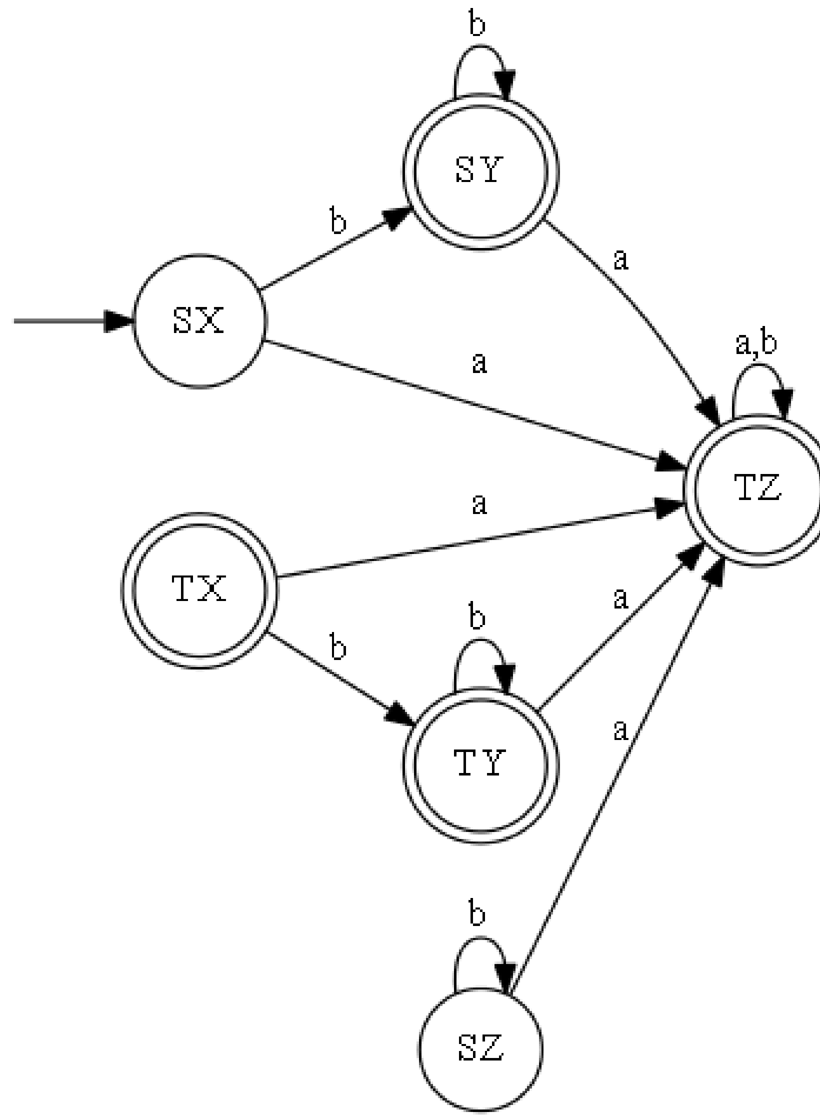
L1 = all strings that include “a”





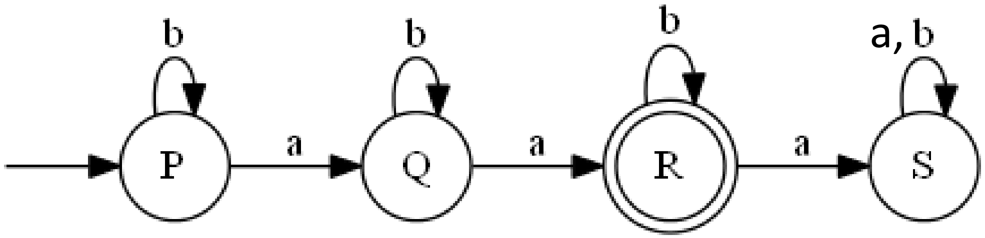
L2 = all strings that do not include “a”

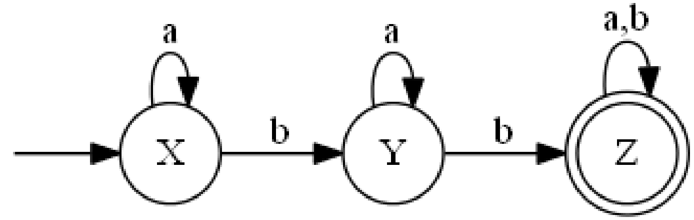
Union of L1 and L2



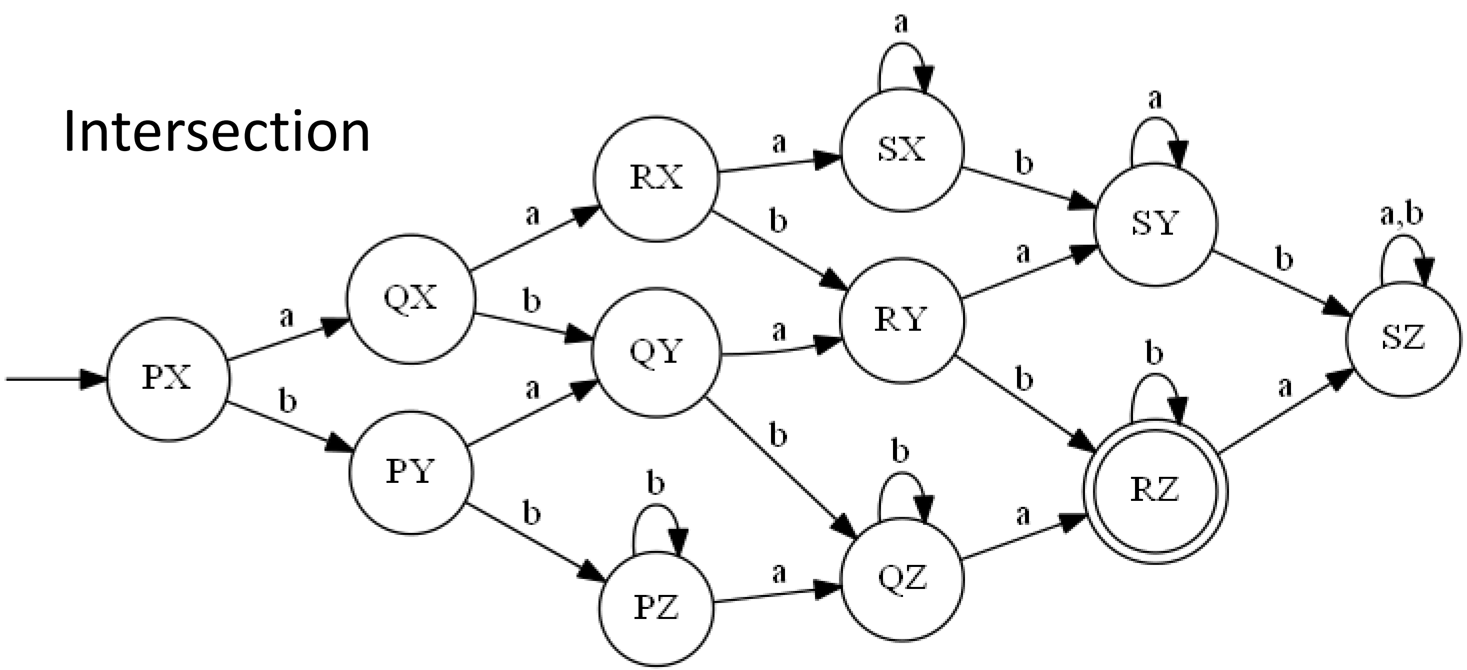
Construct an FA to accept strings with exactly 2 a's and at least 2 b's

Strings with exactly 2 a's $\bigcap$ Strings with at least 2 b's

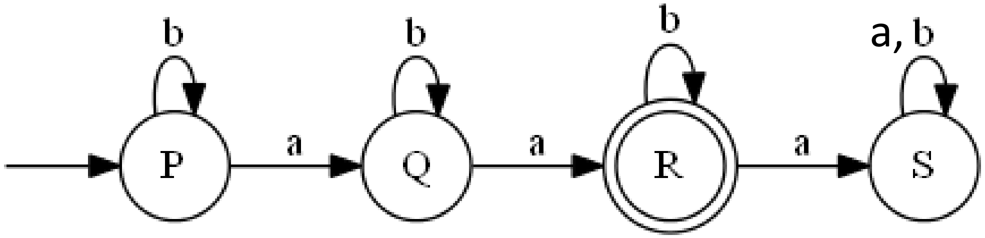




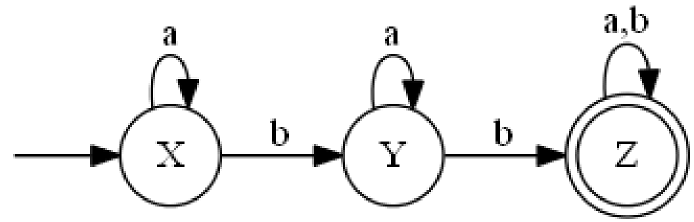
Intersection



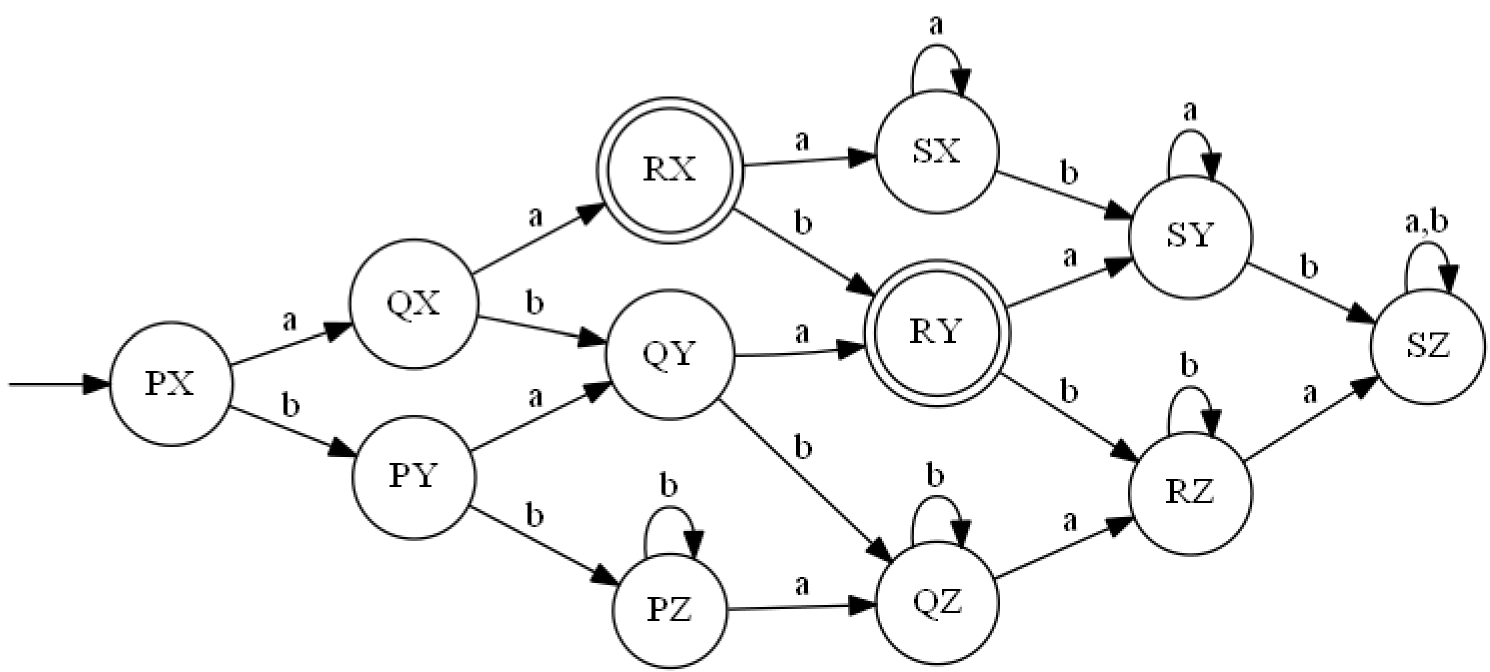
Strings with exactly 2 a’s



Strings with at least 2 b’s



Difference, i.e., strings with exactly 2 a’s and at most 1 b



We prove the first part of the theorem.

- Theorem: Suppose $M_1=(Q_1, \Sigma, q_1, A_1, \delta_1)$ and $M_2=(Q_2, \Sigma, q_2, A_2, \delta_2)$ are FAs accepting L_1 and L_2. Let $M=(Q, \Sigma, q_0, A, \delta)$ be defined as follows:
  - $Q = Q_1 \times Q_2$
  - $q_0 = (q_1, q_2)$
  - $\delta((p, q), \sigma) = (\delta_1(p, \sigma), \delta_2(q, \sigma))$

then, if $A = \{(p, q) \mid p \in A_1 \text{ or } q \in A_2\}$, M accepts $L_1 \cup L_2$.

Proof sketch. At any point during the operation of M, if (p, q) is the current state, then p, and q are the current states of M_1, and M_2. This follows from

Prove by structural induction at home

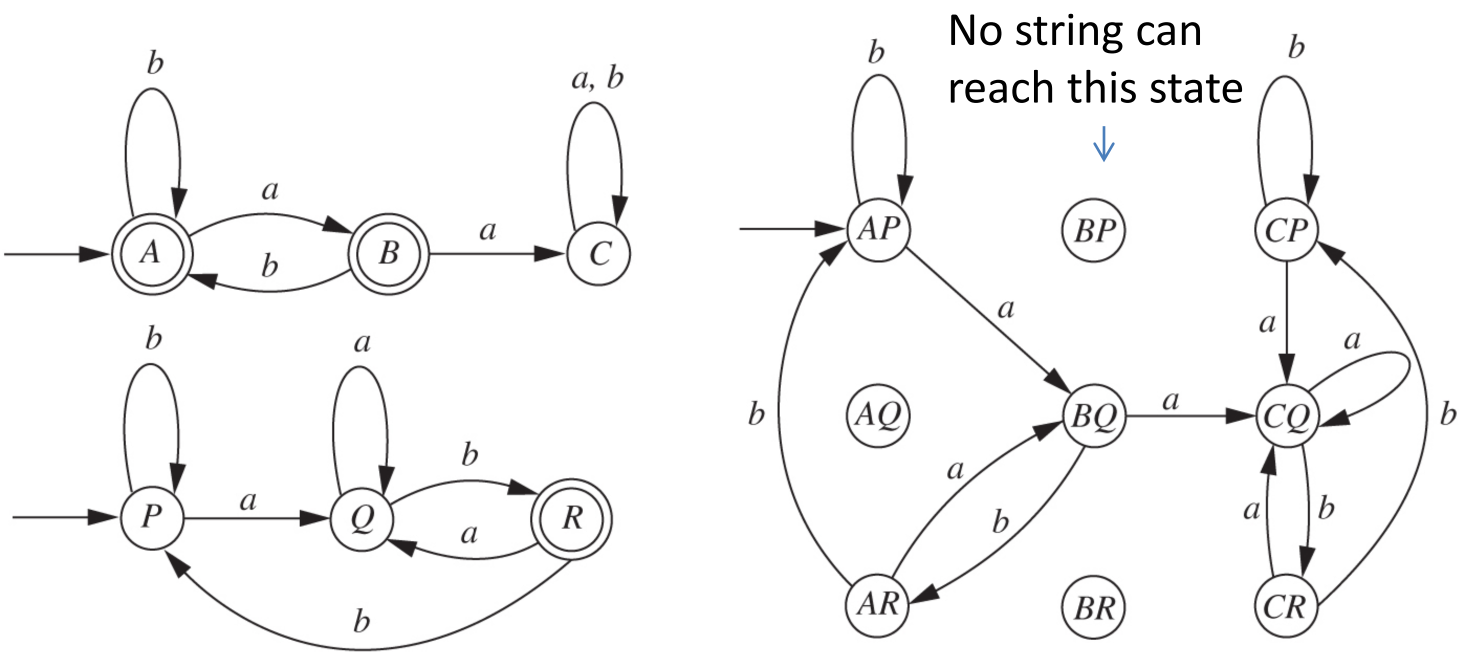
$$\delta^*(q_0, x) = (\delta_1^*(q_1, x), \delta_2^*(q_2, x)) \quad \forall x \in \Sigma^*.$$

For every $x \in \Sigma^*$, x is accepted by M iff $\delta^*(q_0, x) \in A$ but according to defs of A and δ^*, this is true if $\delta_1^*(q_1, x) \in A_1$ or $\delta_2^*(q_2, x) \in A_2$, i.e., if $x \in L_1 \cup L_2$. $\square$

Homework: Make sure you learn how to prove the full version of this theorem including the structural induction, union, intersection and difference (see textbook). Don’t submit!

Accepting the Union, Intersection, or Difference of Two Languages (cont'd.)

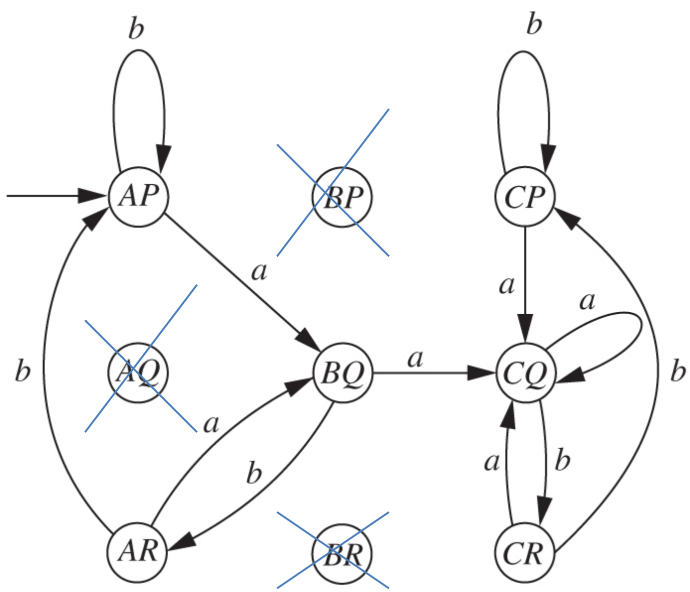
- Given two machines, create the Cartesian product of the state sets, and draw the necessary transitions



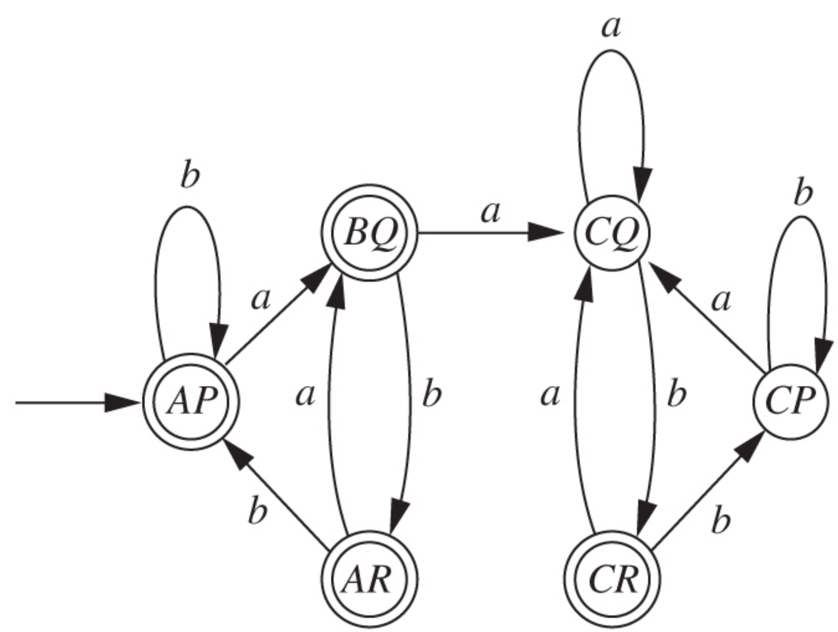
(a) (b)

Accepting the Union, Intersection, or Difference of Two Languages (cont'd.)

- Simplify the resulting machine, if possible, and designate the appropriate accepting states
- The machine below accepts the **union** of the two languages



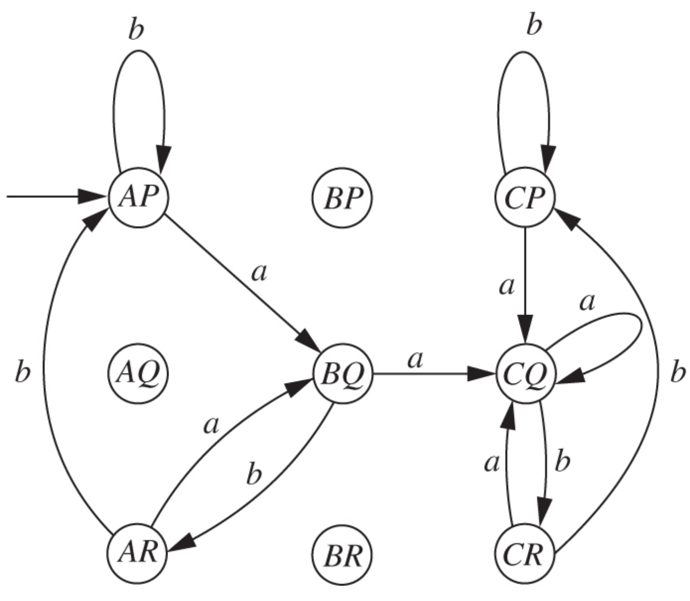
(b)



(c)

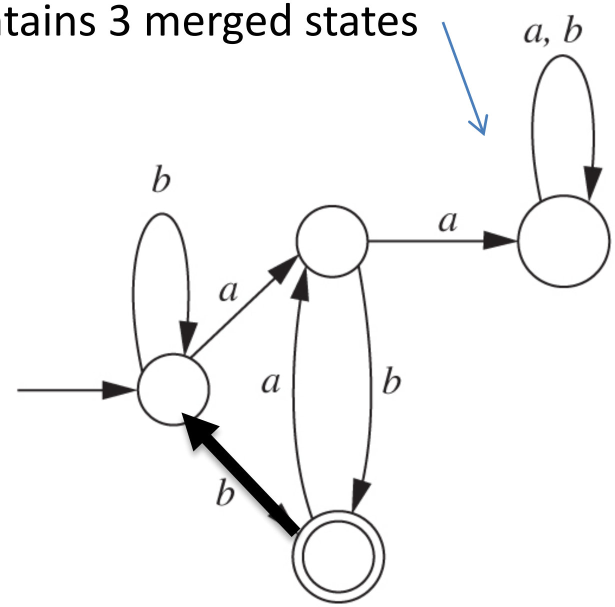
Accepting the Union, Intersection, or Difference of Two Languages (cont'd.)

- For the **<u>intersection</u>**, we can simplify further, and we end up with the machine on the right
- The simplification involved turning states CP, CQ, and CR into a single state (none of them was accepting, and there was no way to leave them)



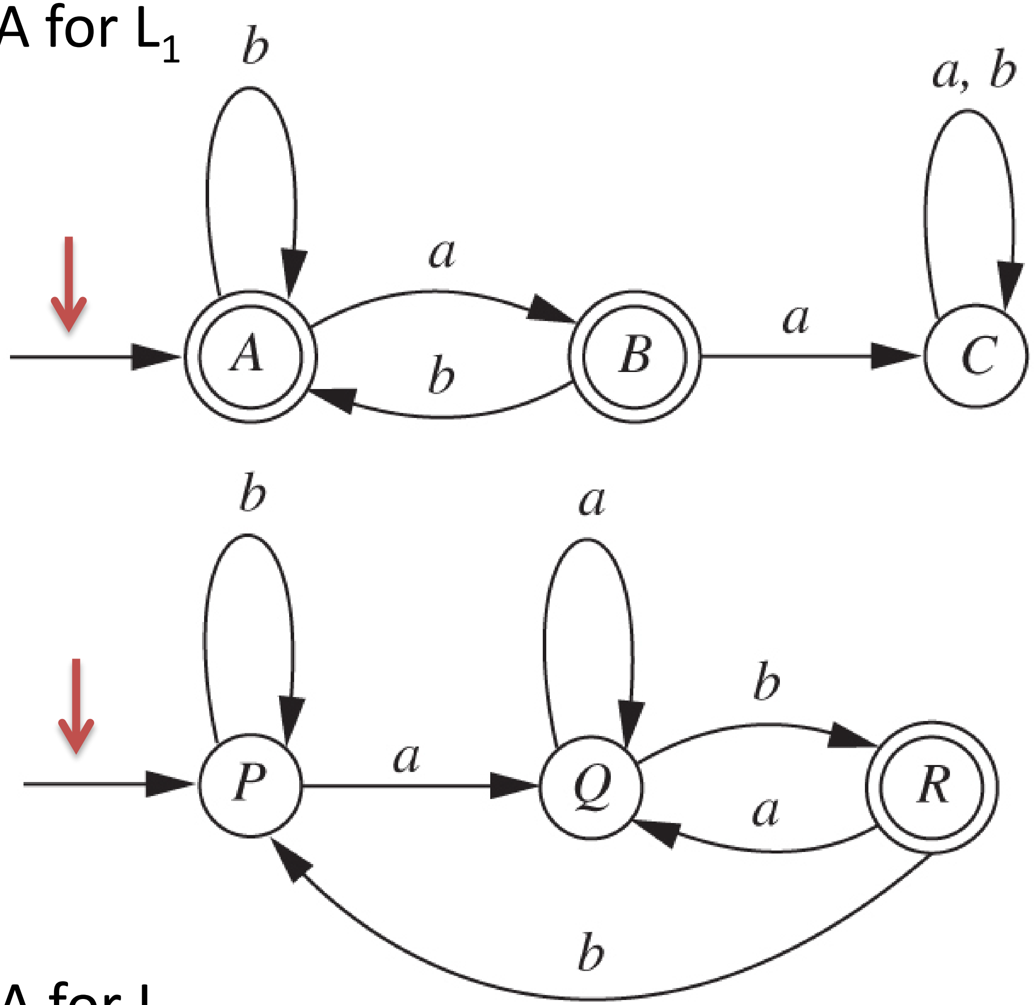
(b)

Contains 3 merged states



(d)

FA for L_1



FA for L_2

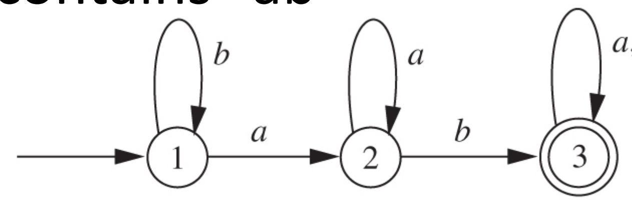
(a)

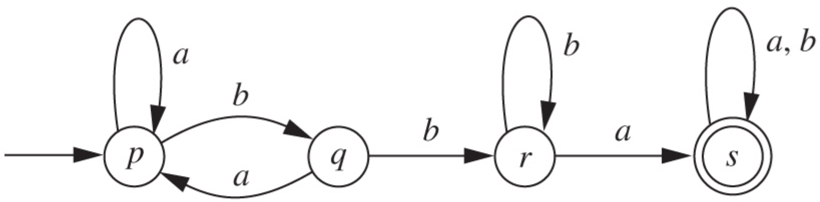
Input: abbaa

		$L_1 \cup L_2$	$L_1 \cap L_2$	$L_1 - L_2$
_	AP	y	n	y
a	BQ	y	n	y
b	AR	y	y	n
b	AP	y	n	y
a	BQ	y	n	y
a	CQ	n	n	n

Example: FA accepting strings that contain either “ab” or “bba”

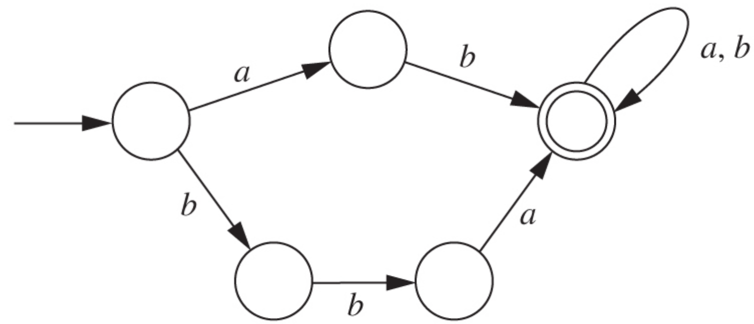
x contains “ab”



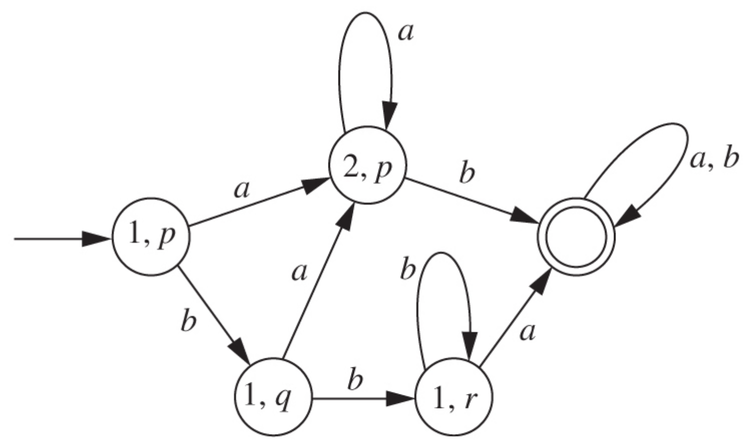


x contains “bba” (a

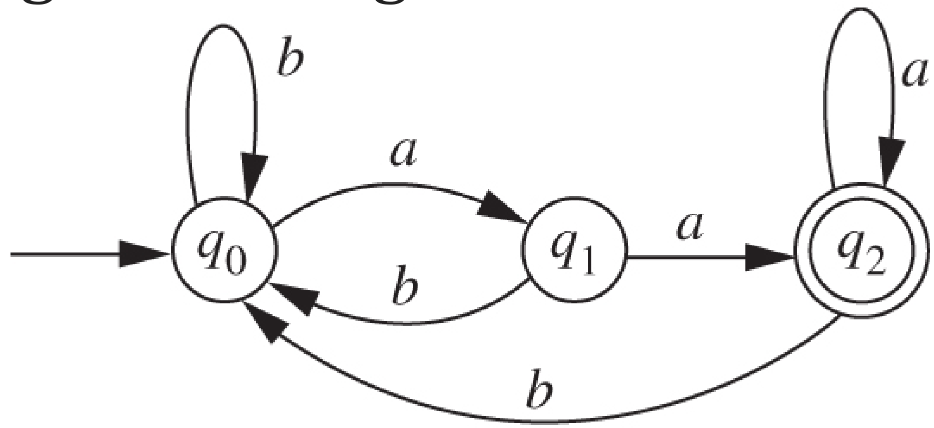
We can try to build FA with 12 states but we can do less with “if”-like fork



(b)



Distinguishing One String from Another



Strings ending in *aa*

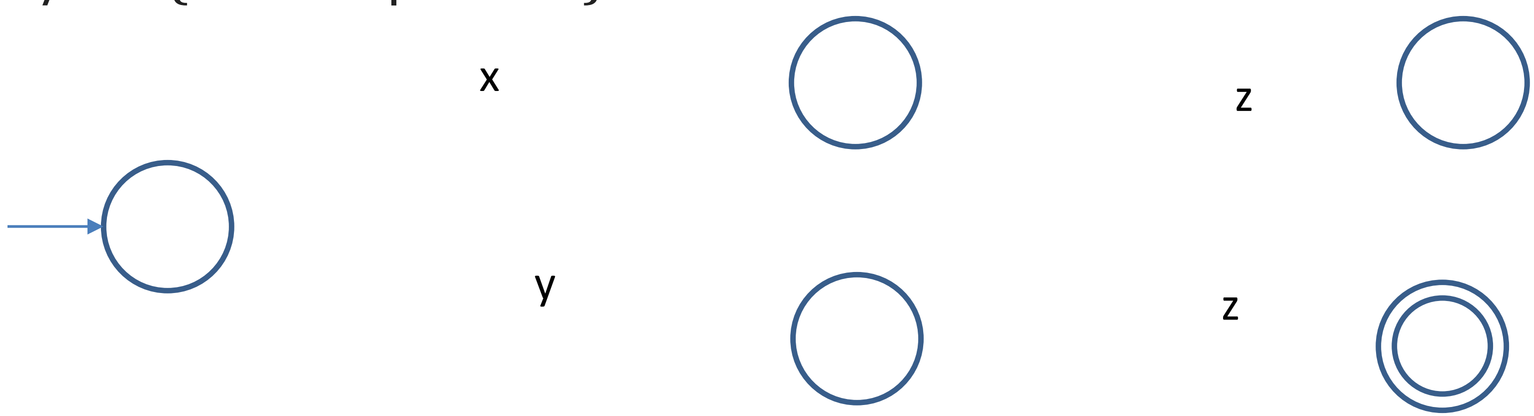
- Any FA, ignores, or "forgets", a lot of information
- An FA doesn't remember which string has been seen

  - *aba* and *aabbabbabaaaba* lead to the <u>same state</u>;
  - *aba* and *ab*, however, lead to <u>different states</u>; the essential difference is that one ends with *a* and the other doesn't
  - *aba* and *ab* are *distinguishable* with respect to the language accepted by the FA; there is at least one string *z* (such as *a*) so that *abaz* is in the language (i.e., is accepted) and *abz* is not, or vice versa

Distinguishing One String from Another (cont'd.)

- Definition:
  - If L is a language over Σ, and $x, y \in \Sigma^*$, then x and y are *L-distinguishable*, if there is a string $z \in \Sigma^*$ such that

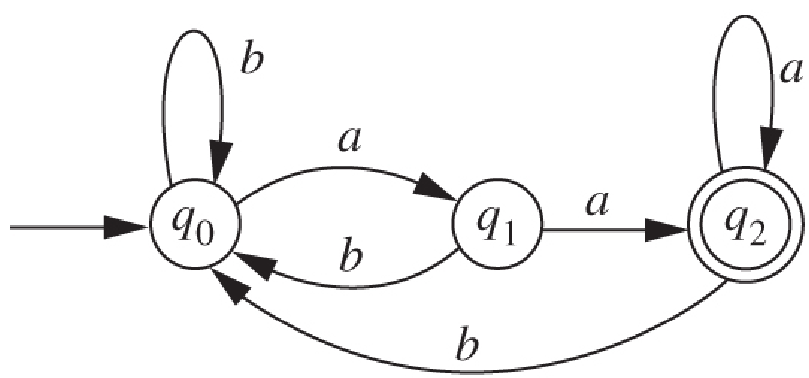
    either $xz \in L$ and $yz \notin L$, or $xz \notin L$ and $yz \in L$
  - A string z having this property is said to distinguish x and y with respect to L
  - Equivalently, x and y are L-distinguishable if $L/x \neq L/y$, where $L/x = \{z \in \Sigma^* \mid xz \in L\}$



Distinguishing One String from Another (cont'd.)

- **Theorem**: Suppose $M=(Q, \Sigma, q_0, A, \delta)$ is an FA accepting $L \subseteq \Sigma^*$
  - If $x, y \in \Sigma^*$ are L-distinguishable, then $\delta^*(q_0, x) \neq \delta^*(q_0, y)$
  - For all $n \geq 2$, if there is a set of n pairwise L-distinguishable strings in Σ^*, then Q must contain **at least** n states

This shows why we need at least three states in any FA that accepts <u>the language L of strings ending in aa</u>: $\{\Lambda, a, aa\}$ contains 3 pairwise L-distinguishable strings



Part I: If x and y are two strings in Σ^* that are L-distinguishable, then $\delta^*(q_0, x) \neq \delta^*(q_0, y)$

Proof sketch. x, and y are L-distinguishable $\Rightarrow$ $\exists z \in \Sigma^*$ s.t. $xz \in L$, and $yz \notin L$ (or vice versa). Because M accepts L then either $\delta^*(q_0, xz) \in A$ and $\delta^*(q_0, yz) \notin A$ (or vice versa), i.e.,

$$\delta^*(q_0, xz) \neq \delta^*(q_0, yz).$$

However,

$$\begin{aligned} \delta^*(q_0, xz) &= \delta^*(\delta^*(q_0, x), z) \\ \delta^*(q_0, yz) &= \delta^*(\delta^*(q_0, y), z). \end{aligned}$$

Because the left sides are different, the right sides must be also, and then

$$\delta^*(q_0, x) \neq \delta^*(q_0, y).$$

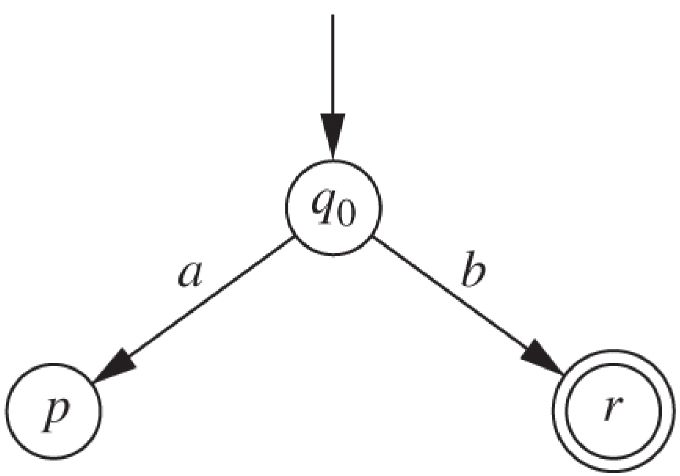
$\square$

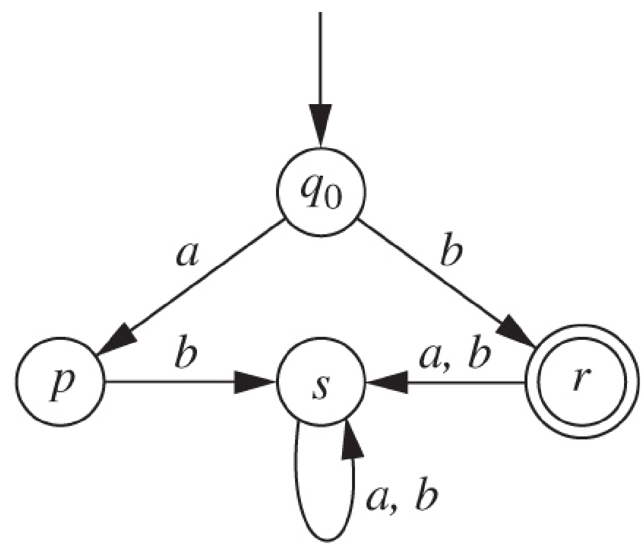
Part II: For all $n \geq 2$, **if** there is a set of n pairwise L-distinguishable strings in Σ^*, **then** Q must contain at least n states

Proof sketch. The second part of the theorem follows from the first. If M had fewer than n states, then at least two of the n strings would cause M to end up in the same state. This is contradiction because these strings are L-distinguishable. $\square$

Distinguishing One String from Another (cont'd.)

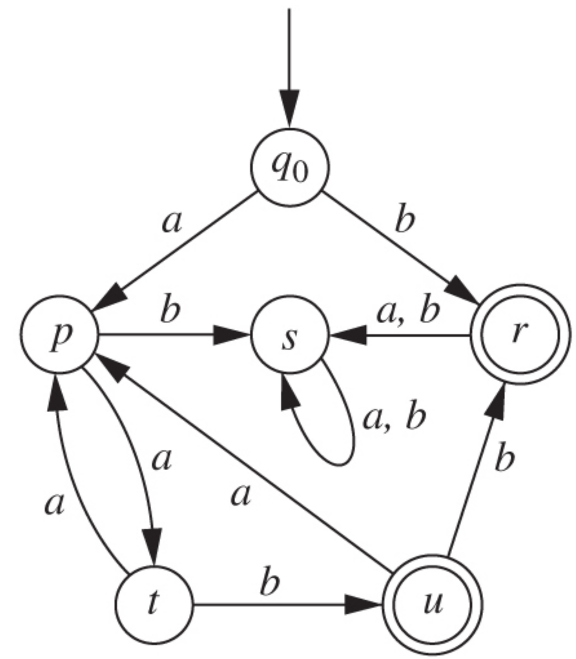
- To create an FA to accept $L = L_1L_2 = \{aa, aab\}^*\{b\}$, we notice first that Λ,a $\notin L$, $b \in L$, and Λ,b, and a are L-distinguishable (for example, $\Lambda b \in L$, $ab \notin L$)
  - We need at least the states in the first diagram
  - L contains b but nothing else that begins with b, so we add a state s to take care of illegal prefixes
  - If the input starts with aa we, need to leave state p because a and aa are L-distinguishable; create state t





Distinguishing One String from Another (cont'd.)

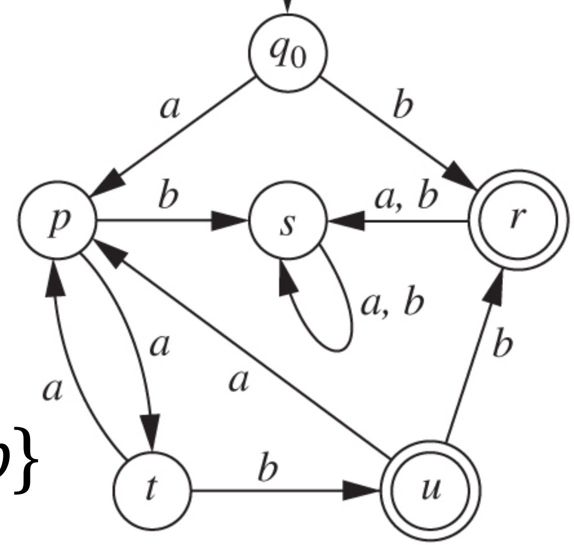
- $\delta(t, b)$ must be accepting, because $aab \in L$ but distinguishable from b; call that new state u
- Let $\delta(u,b)$ be r, because $aabb$ is in L but not a prefix of any other string in L
- States t and u can be thought of as representing the end of an occurrence of aa or aab; if the next symbol is a it's the start of a new occurrence, so go back to p
- The result is shown here



Distinguishing One String from Another (cont'd.)

	Λ	a	b	aab	aa	ab
Λ	-	b	Λ	Λ	bb	b
a		-	Λ	b	b	ab
b			-	aab	bb	Λ
aab				-	bb	Λ
aa					-	b
ab						-

These are strings z's that distinguish between x, and y



FA to accept $L = \{aa, aab\}^*\{b\}$

Theorem. *For every $L \subseteq \Sigma^*$, if there is an infinite set S of pairwise L-distinguishable strings, then L cannot be accepted by FA.*

Proof sketch. If S is infinite then for every n, S has a subset with n elements. If M was FA accepting L, then previous theorem would say that for every n, M would have at least n states. No FA has this property. $\square$