Chapter 9

Regular Languages

9.1 Properties of Regular Languages

Definition: A language that can be defined by a regular expression is a *regular language.*

Theorem 10 *If L_1 and L_2 are regular languages, then $L_1 + L_2$, L_1L_2, and L_1^* are also regular languages.*

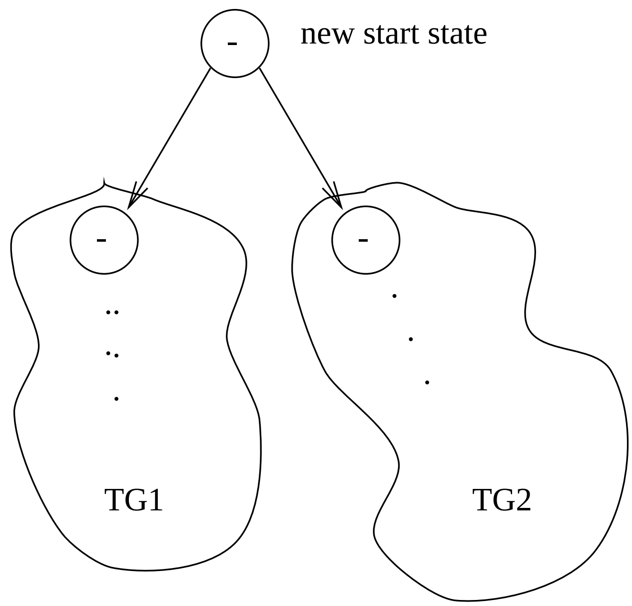
Proof. (by regular expressions)

- If L_1 and L_2 are regular languages, then there are regular expressions $\mathbf{r}_1$ and $\mathbf{r}_2$ that define these languages.
- $\mathbf{r}_1 + \mathbf{r}_2$ is a regular expression that defines the language $L_1 + L_2$, and so $L_1 + L_2$ is a regular language.
- $\mathbf{r}_1\mathbf{r}_2$ is a regular expression that defines the language L_1L_2, and so L_1L_2 is a regular language.
- $\mathbf{r}_1^*$ is a regular expression that defines the language L_1^*, and so L_1^* is a regular language.

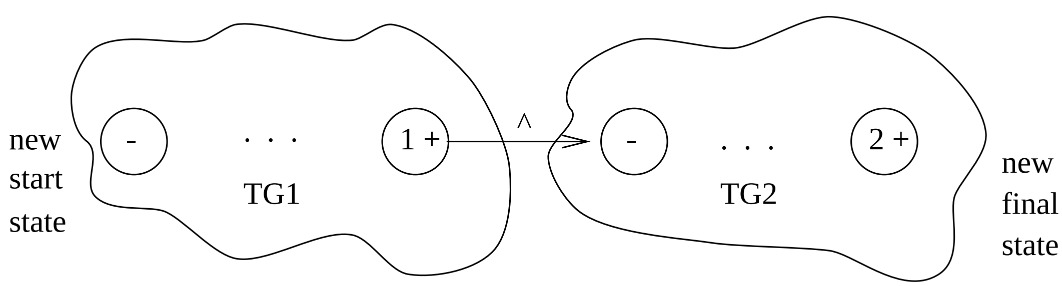
∎

Proof. (by machines)

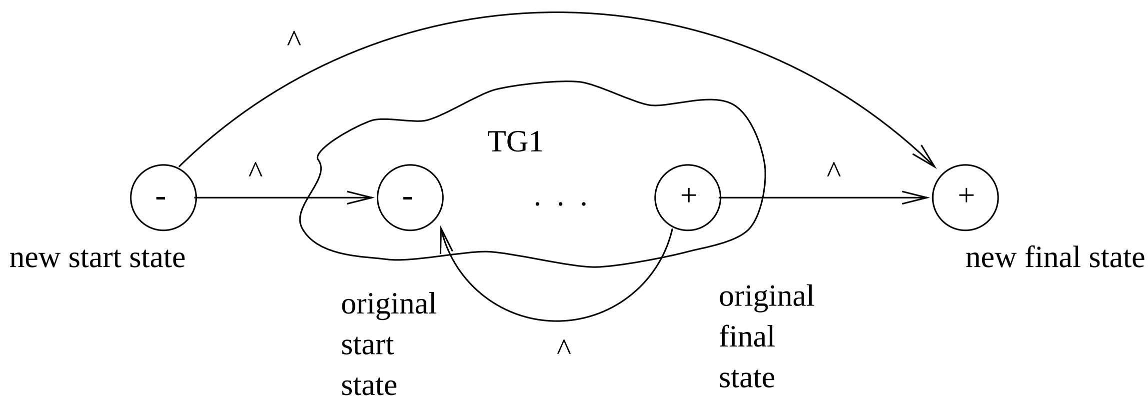
- If L_1 and L_2 are regular languages, then there are transition graphs TG_1 and TG_2 that accept them by Kleene's Theorem.
- We may assume that TG_1 has a unique start state and unique final state, and the same for TG_2.
- We construct the TG for $L_1 + L_2$ as follows:



- We construct the TG for L_1L_2 as follows:

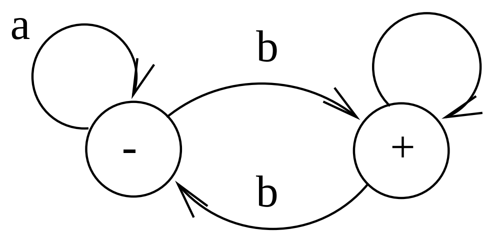


- We construct the TG for L_1^* as follows:

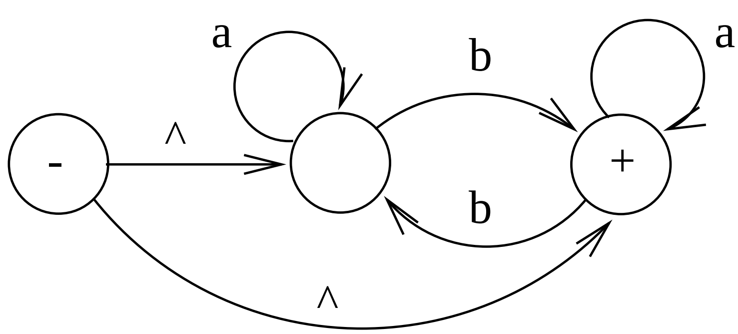


Remarks:

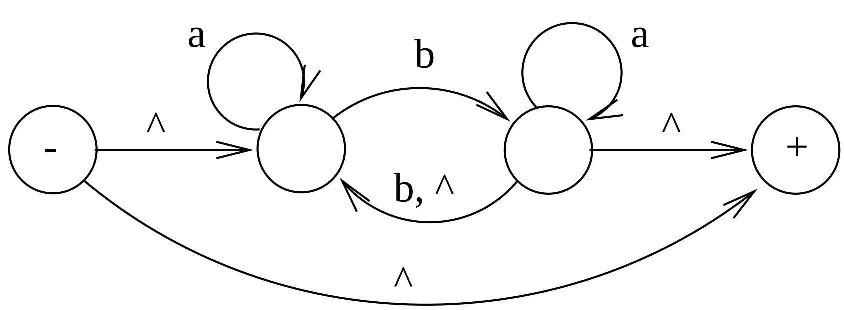
- The technique given in the tapes of lectures 11 and 12 is wrong.
- To see why, consider the following FA for the language
$L = \{$ words having an odd number of b's $\}$



- Note that L^* is the language consisting of Λ and all words having at least one b, which has regular expression $\mathbf{\Lambda} + (\mathbf{a}+\mathbf{b})^*\mathbf{b}(\mathbf{a}+\mathbf{b})^*$ (which was also wrong in the tape of lecture 12).
- If we use the (incorrect) technique to construct a TG for L^* given in the taped lecture, then we get the following:



- However, the above TG accepts the string $a \notin L^*$.
- On the other hand, if we use the method presented above to construct a TG for L^*, then we get the following correct TG:

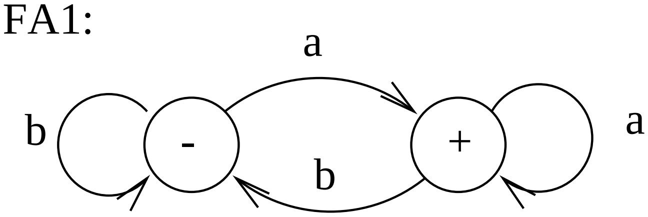


■

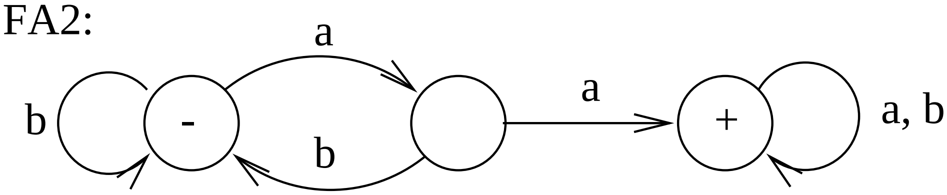
Example: alphabet $\Sigma = \{a, b\}$
L_1 = all words ending with a
L_2 = all words containing the substring aa.

Regular expressions:
$\mathbf{r}_1 = (\mathbf{a} + \mathbf{b})^*\mathbf{a}$
$\mathbf{r}_2 = (\mathbf{a} + \mathbf{b})^*\mathbf{aa}(\mathbf{a} + \mathbf{b})^*$

FA1:

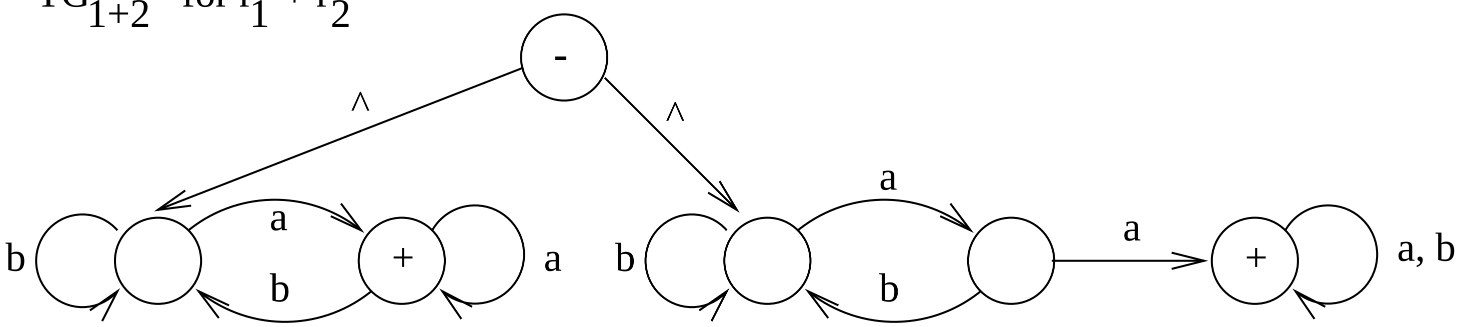


FA2:

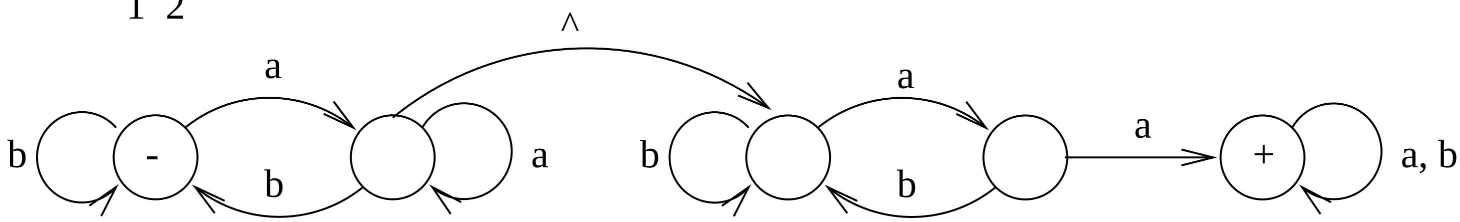


$r_1 + r_2 = (a+b)^*a + (a+b)^*aa(a+b)^*$

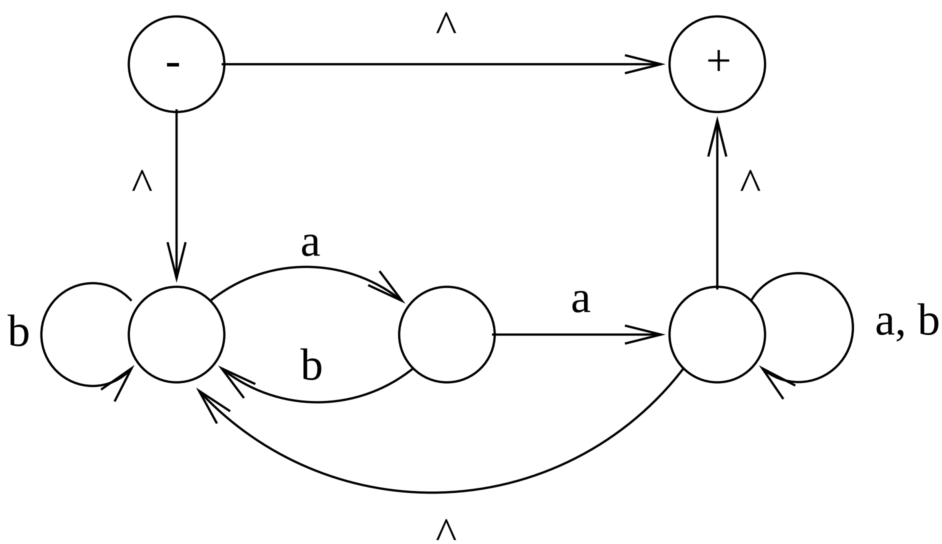
TG_{1+2} for $r_1 + r_2$



TG for $r_1 r_2$



TG for r_2^*



9.2 Complementation of Regular Languages

Definition: If L is a language over the alphabet Σ, we define L' to be its *complement*, which is the language of all strings of letters from Σ that are not words in L, i.e., $L' = \{w \in \Sigma^* : w \notin L\}$.

Example: alphabet $\Sigma = \{a, b\}$
L = language of all words in Σ^* containing the substring *abb*.
L' = language of all words in Σ^* not containing the substring *abb*.

Note that

$$(L')' = L$$

Theorem 11 *If L is a regular language, then L' is also a regular language. In other words, the set of regular languages is closed under complementation.*

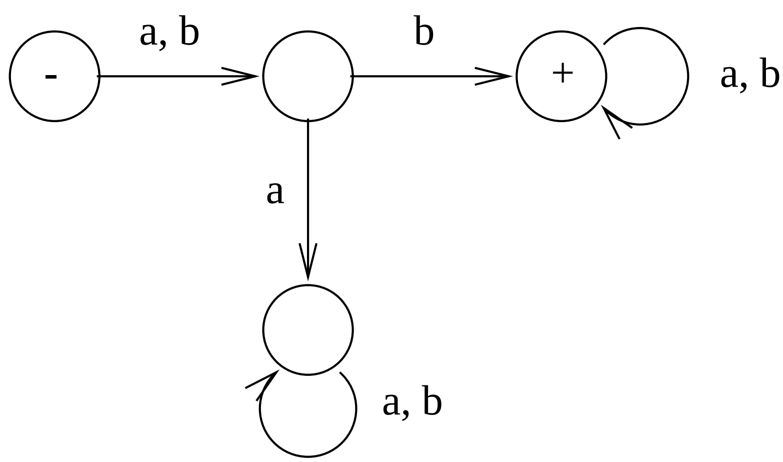
Proof.

- If L is a regular language, then there exists some FA that accepts L by Kleene's Theorem.
- Create new finite automaton FA' from FA as follows:
  - FA' has same states and arcs as FA.
  - Every final state of FA becomes a nonfinal state in FA'
  - Every nonfinal state of FA becomes a final state in FA'
  - FA' has same start state as FA.
- FA' accepts the language L'.
- Kleene's Theorem implies that L' is a regular language.

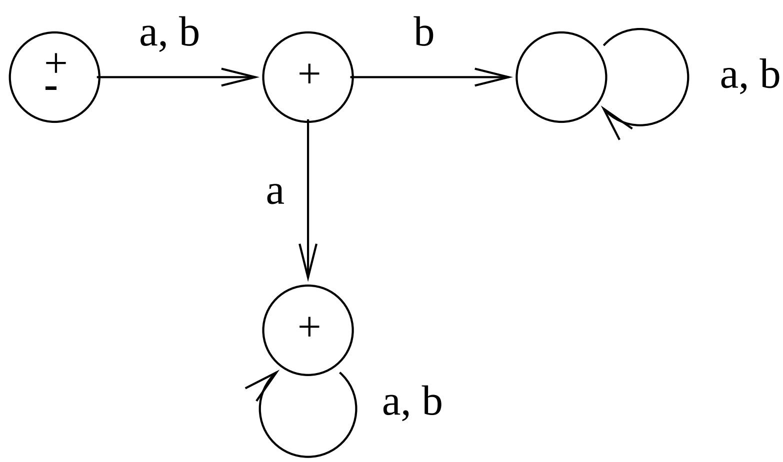
∎

Example: $\Sigma = \{a, b\}$
L = all words with length at least 2 and second letter b
L' = all words with length less than 2 or second letter a

FA:



FA':



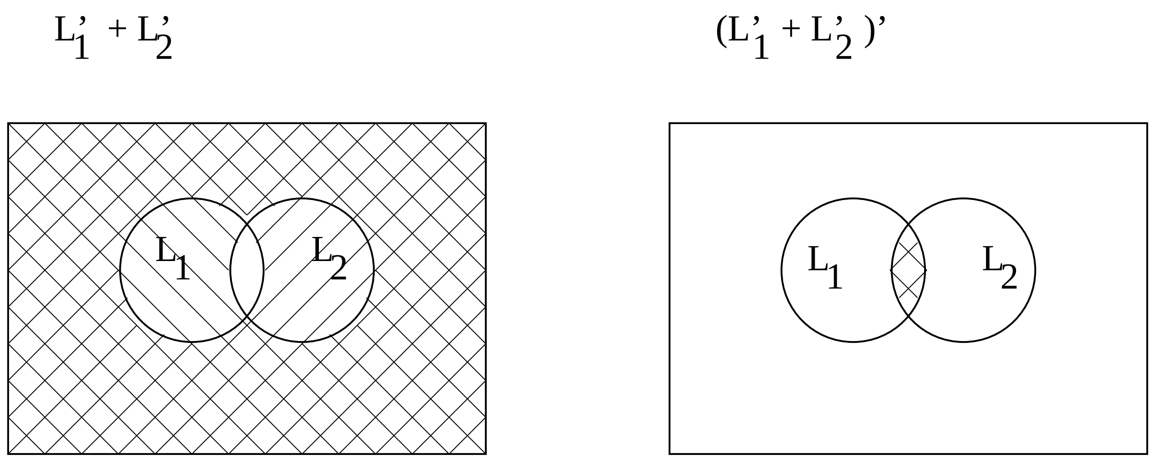
9.3 Intersections of Regular Languages

Theorem 12 *If L_1 and L_2 are regular languages, then $L_1 \cap L_2$ is a regular language. In other words, the set of regular languages is closed under intersection.*

Proof.

- DeMorgan's Law for sets states that

$$L_1 \cap L_2 = (L_1' + L_2')'$$

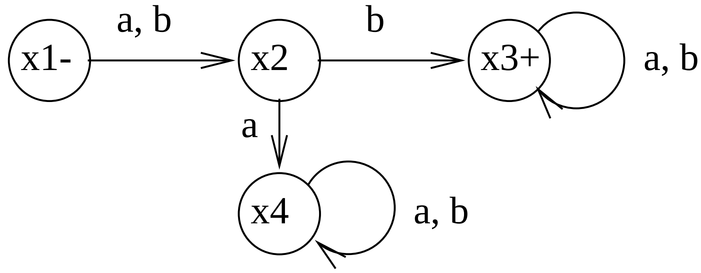


- Since L_1 and L_2 are regular languages, Theorem 11 implies that L_1' and L_2' are regular languages.
- Theorem 10 then implies that $L_1' + L_2'$ is a regular language.
- Theorem 11 then implies that $(L_1' + L_2')'$ is a regular language.

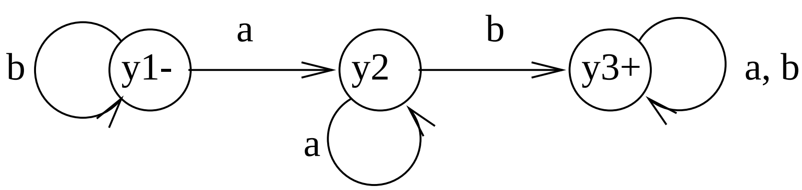
Example: alphabet $\Sigma = \{a, b\}$
L_1 = all words with length ≥ 2 and second letter b
L_2 = all words containing the substring ab.

r1 = (a+b)b(a+b)*   r2 = (a+b)*ab(a+b)*

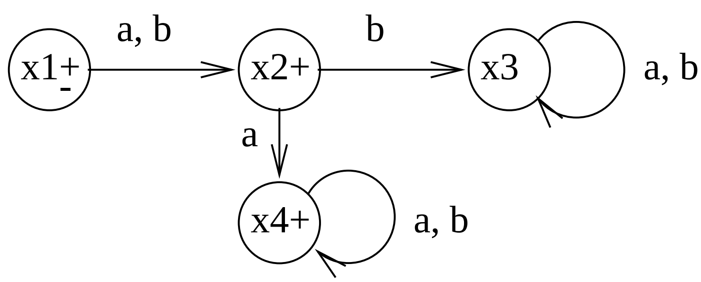
FA1 :



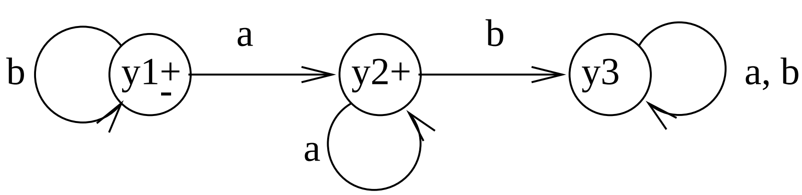
FA2 :



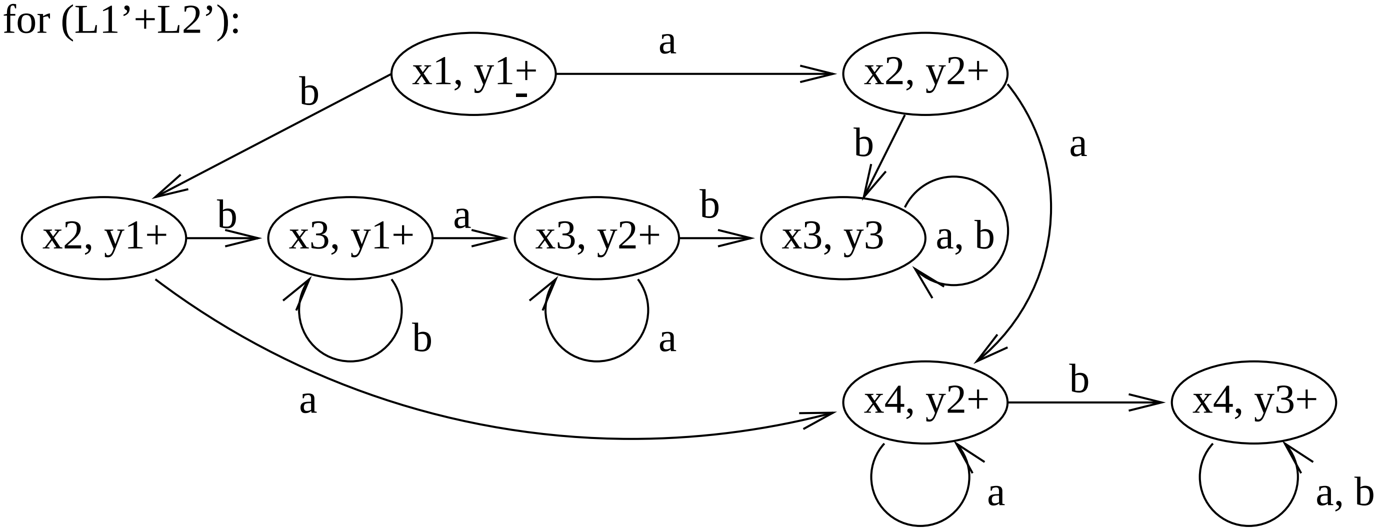
FA1' :



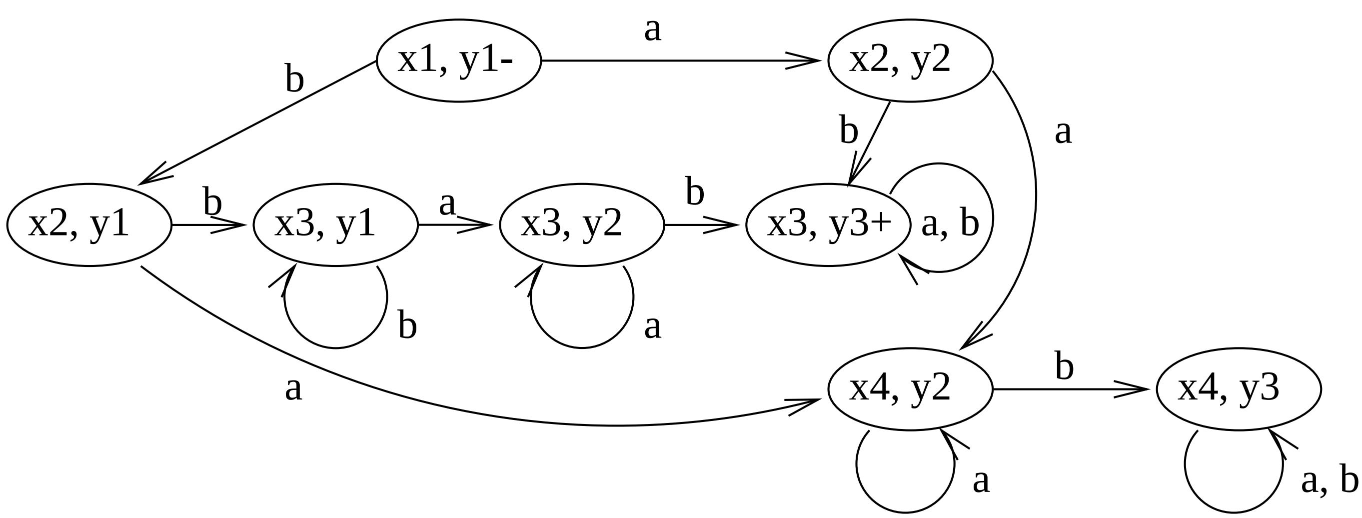
FA2' :



FA for (L1'+L2'):



FA for (L1'+L2')' :



As an exercise, we will now derive a regular expression for $L_1 \cap L_2$ using our FA for $(L_1' + L_2')'$ and our algorithm from Kleene's theorem:

Proof. (another for Theorem 12)

- In proof of Kleene's theorem, we showed how to construct FA_3 that is the union of FA_1 and FA_2.
- Suppose states of FA_1 are $x_1, x_2, \ldots$.
- Suppose states of FA_2 are $y_1, y_2, \ldots$.
- We do the same construction of FA_3 except we now make a state in FA_3 a final state only if both the corresponding x and y states are final states.
- Then FA_3 accepts only words that are accepted by both FA_1 and FA_2.

■

Example: alphabet $\Sigma = \{a, b\}$
L_1 = all words with length ≥ 2 and second letter b
L_2 = all words containing the substring ab.