Conversions Among FAs and REs

We describe algorithms that convert among NFAs, DFAs, and REs.

Kleene's Theorem Revisited

Surprisingly perhaps, nondeterminism does not add to the power of a finite automaton:

Kleene's Theorem. *The following are equivalent for a language L:*

(1) There is a DFA for L.

(2) There is an NFA for L.

(3) There is an RE for L.

This theorem is proved in three conversion algorithms:

$$(3) \Longrightarrow (2) \Longrightarrow (1) \Longrightarrow (3).$$

Conversion From RE to NFA: Recursion

Conversion from RE to NFA uses a recursive construction.

Converting from RE to NFA.

0) If RE empty string, then output simple NFA.

1) If RE single symbol, then output simple NFA.

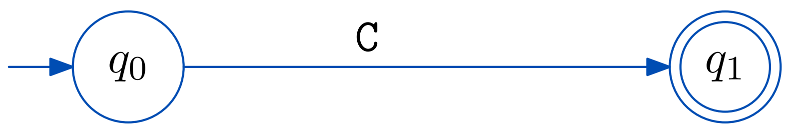
2) If RE has form $A+B$, then combine NFAs for A and B.

3) If RE has form AB, then combine NFAs for A and B.

4) If RE has form A^*, then extend NFA for A.

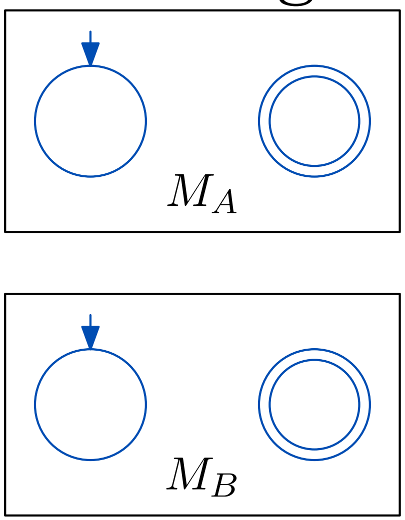
An NFA for a Single Char

An NFA for a single symbol `c` consists of two states:

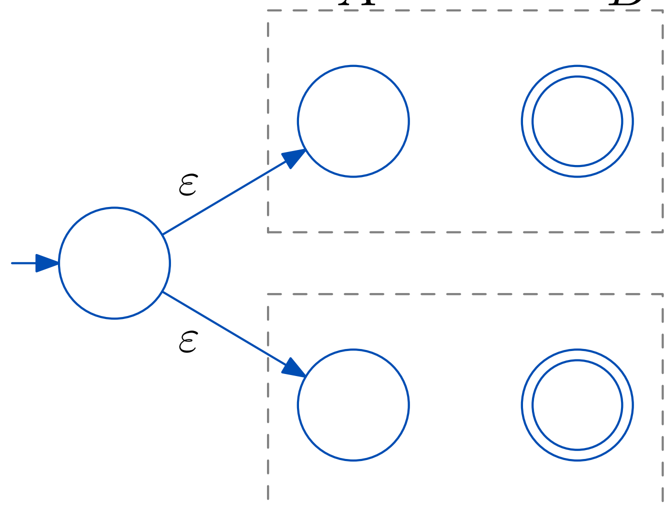


The Union of Two NFAs

Given NFA M_A for A and M_B for B, here is one for $A + B$. Add new start state with ε-transitions to the original start states of both M_A and M_B.



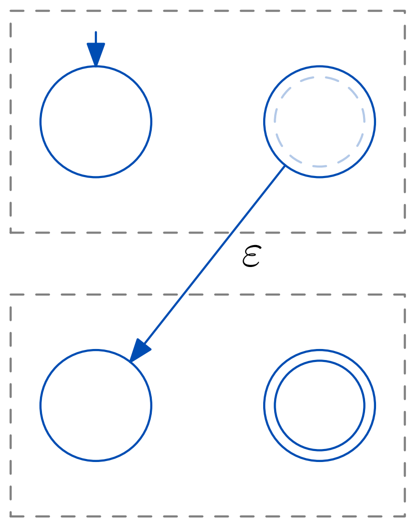
becomes



The machine guesses which of A or B the input is in.

The Concatenation of Two NFAs

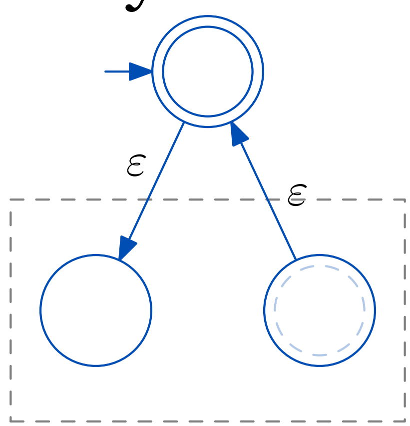
Here is one for AB. Start with NFAs M_A and M_B. First, put ε-transitions from accept states of M_A to start state of M_B. Then make original accept states of M_A reject.



The Star of an NFA

Here is one for A^*. The idea is to allow the machine to cycle from the accept state back to the start state; but we have to be careful.

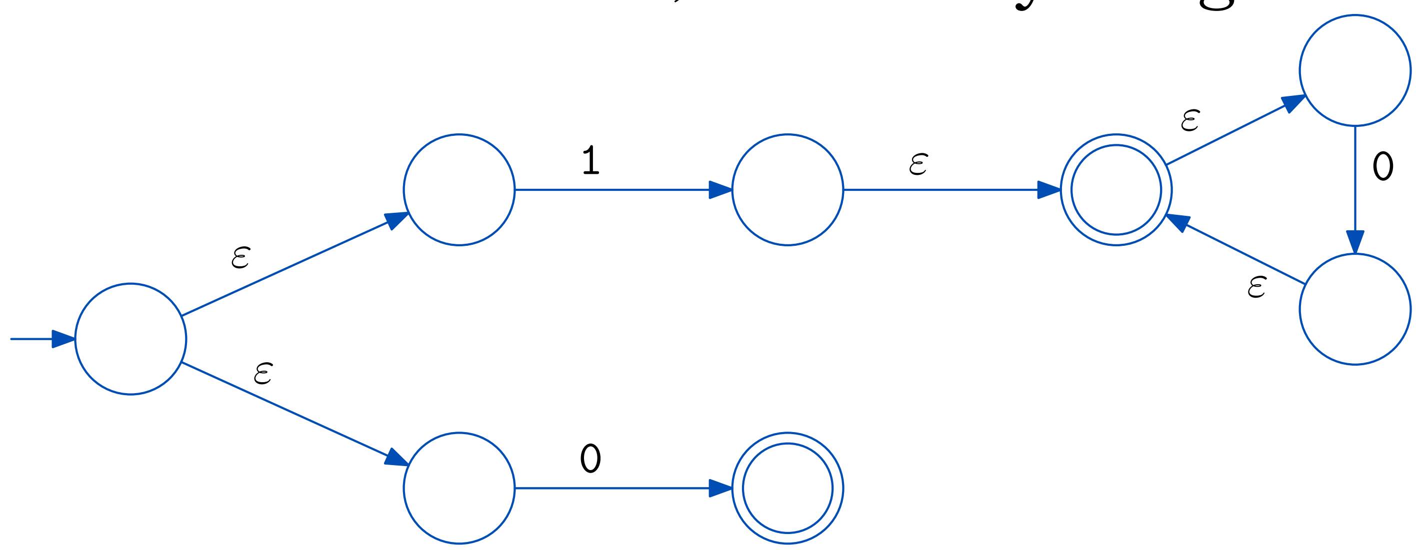
One way to go: build a new start state, which is the only accept state; then put ε-transition from it to old start state, and from old accept states to it; and change every old accept state to reject.



Example of Algorithm

Consider $0 + 10^*$

Build NFAs for the 0, the 1 and the 0^*, then combine the latter two, and finally merge.



Note that resulting NFA can easily be simplified.

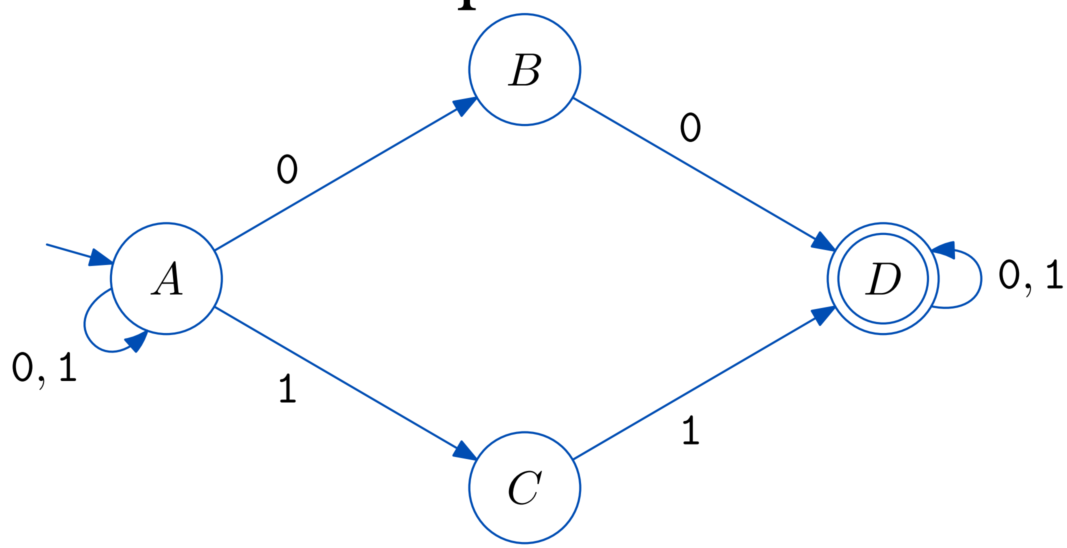
From NFA to DFA: Subset Construction

Converting from NFA to DFA uses the ***subset construction***.

The idea is that to efficiently simulate an NFA on a string, one should at each step keep track of the ***set of states*** the NFA could be in. Note that one can determine the set at one step from the set at the previous step.

Example: Simulating an NFA

Consider `10100` as input to this NFA:



$$\{A\} \xrightarrow{1} \{A, C\} \xrightarrow{0} \{A, B\} \xrightarrow{1} \{A, C\} \xrightarrow{0} \{A, B\} \xrightarrow{0} \{A, B, D\}$$

So accepts `10100` because it can be in accept state after reading the final symbol.

Conversion from NFA to DFA

From NFA (without ε-transitions) to DFA.

0. Each state given by set of states from original.

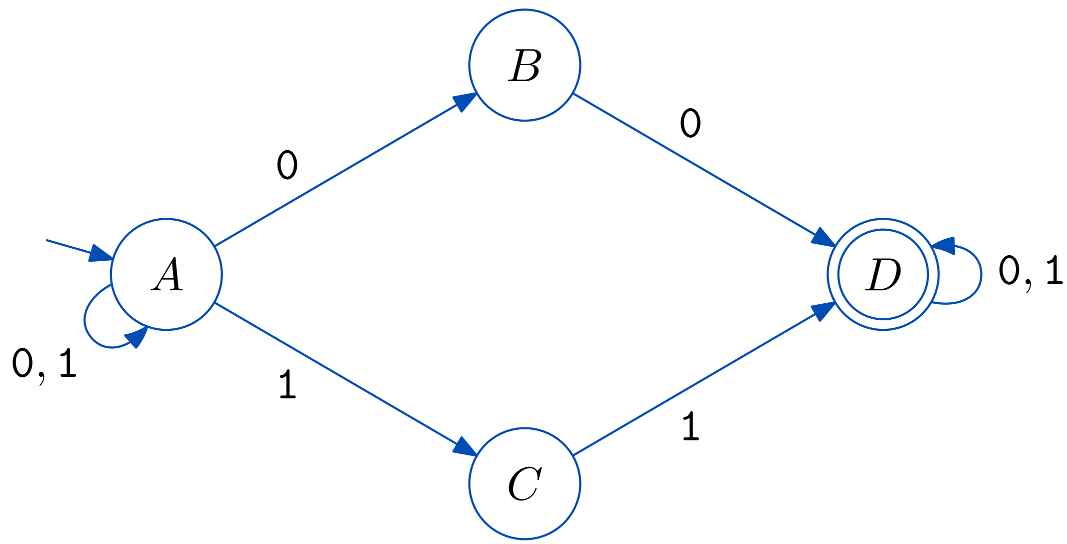
1. Start state is labeled $\{q_0\}$ where q_0 was original start state.

2. While (some state of DFA is missing a transition) do:

compute transition by combining the possibilities for each symbol in the set.

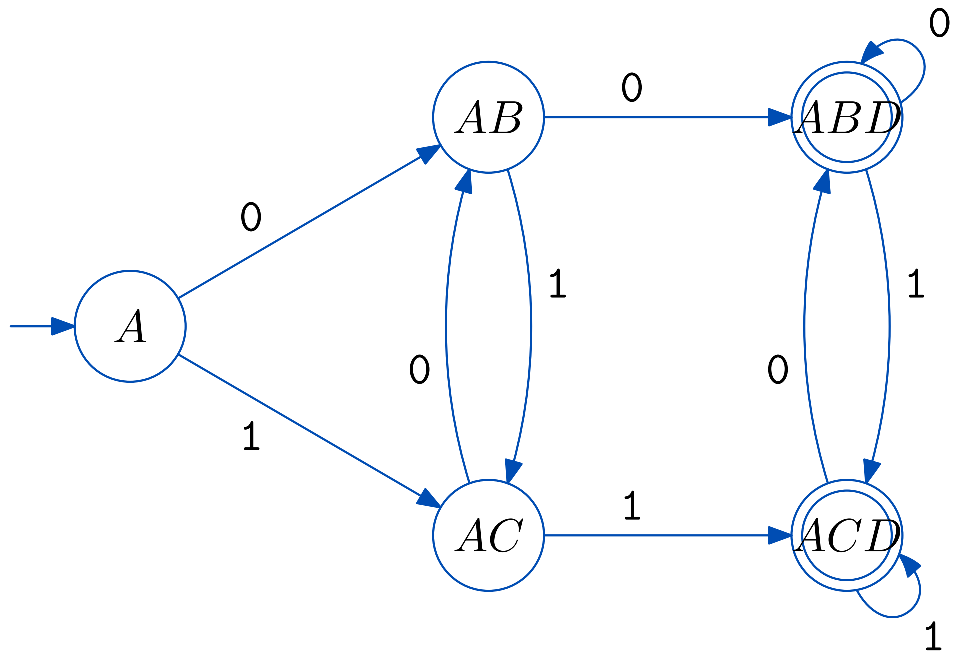
3. Make into accept state any set that contains an original accept state.

Example NFA



Suppose state of DFA was given by the set $\{A, B, D\}$. On 1, the NFA if in state A can go to states A or C, if in state B dies, and if in state D stays in state D. Thus on a 1 the DFA goes to $\{A, C, D\}$. This is an accept state of DFA because it has D.

And the DFA is



Need Closure for ε-Transitions

ε-transitions add a bit more work:

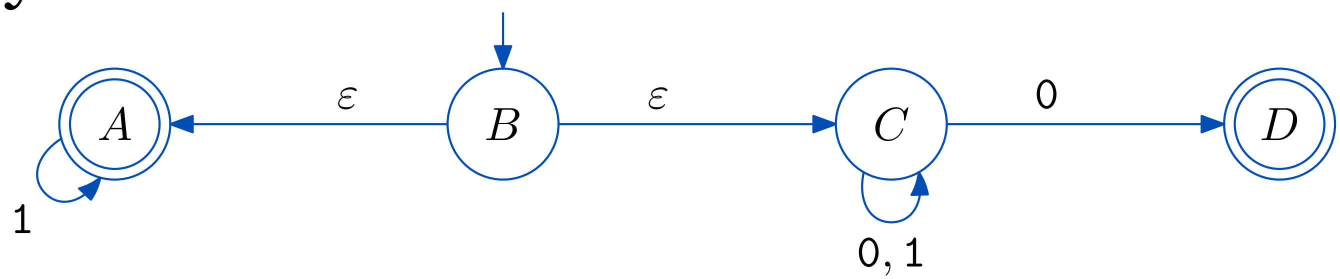
Conversion from NFA (with ε-transitions) to DFA. *As before, except:*

1) The start state becomes the old start state and every state reachable from it by ε-transitions.

*2) When one calculates the states reachable, one includes all states reachable by ε-transitions **after** the destination state.*

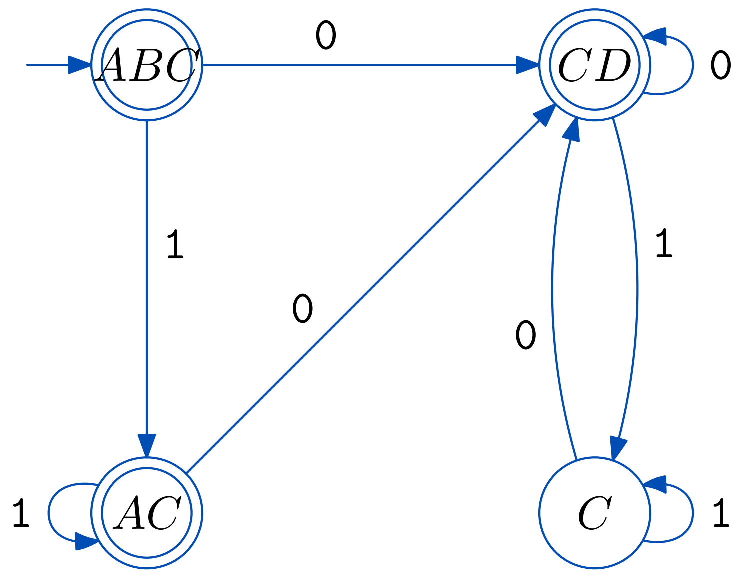
Example NFA

Recall the NFA that accepts all binary strings where the last symbol is 0 or which contain only 1's:



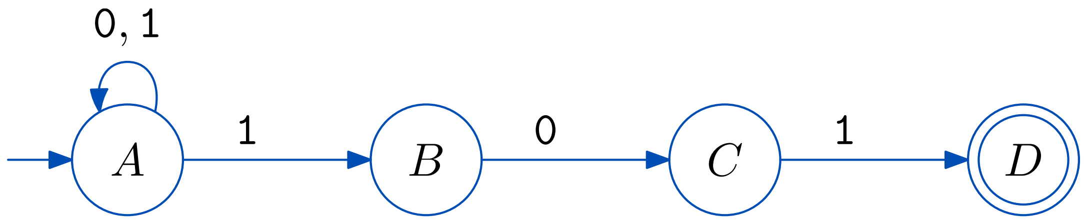
We apply the subset construction. . .

And the DFA is

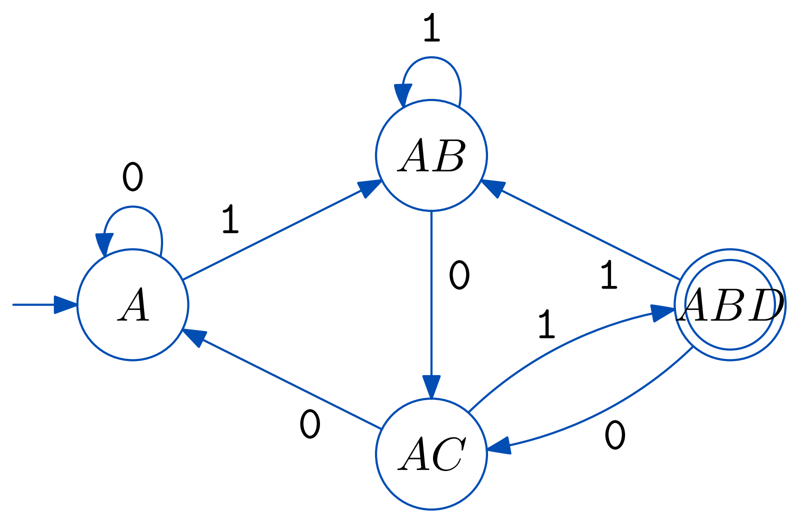


Practice

Convert the following NFA to a DFA using the subset construction:



Solution to Practice



Conversion From FA to RE

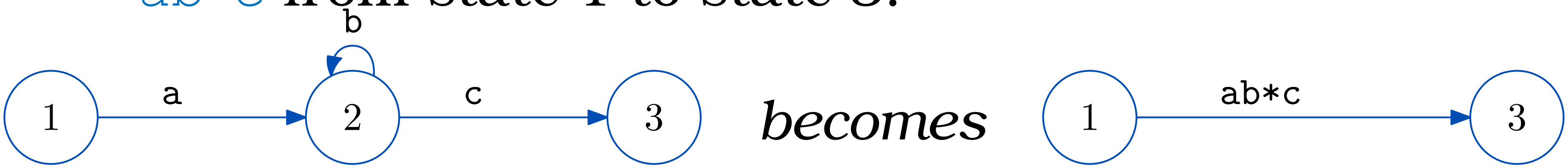
Finally, we show how to convert from FA to RE. One approach is to use a ***generalized FA*** (GFA): each transition is given by an RE.

We build a series of GFAs. At each step, one state (other than start or accept) is removed and replaced by transitions that have the same effect.

Removing a State

Say there is transition a from state 1 to state 2, transition b from state 2 to state 2 and transition c from state 2 to state 3.

One can achieve the same effect by a transition ab^*c from state 1 to state 3.



One must consider all transitions in and out of state 2 simultaneously.

Conversion From FA to RE: Summary

We assume unique accept state, no transition out of accept state, no transition into start state.

Conversion from FA to RE.

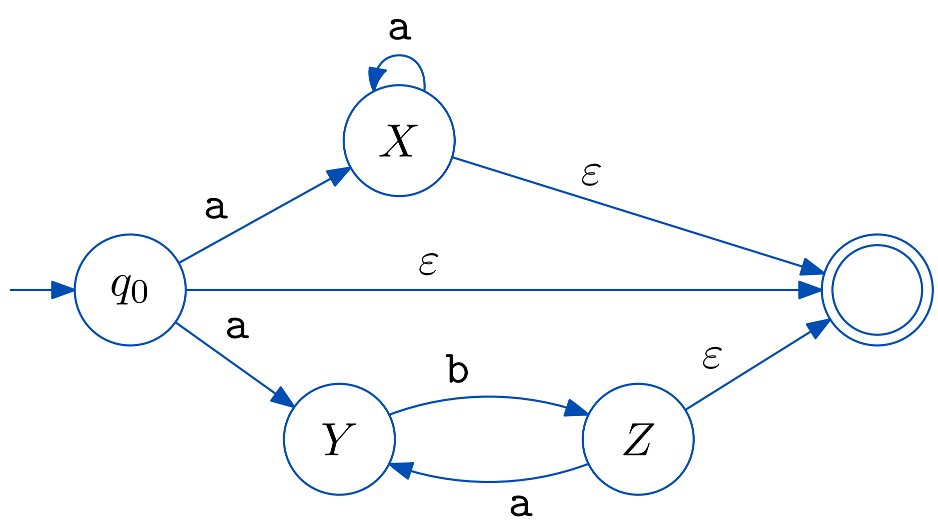
0. Convert to FA of right form.

1. While (more than two states) do
remove one state and replace by appropriate transitions.

2. Read RE off the remaining transition.

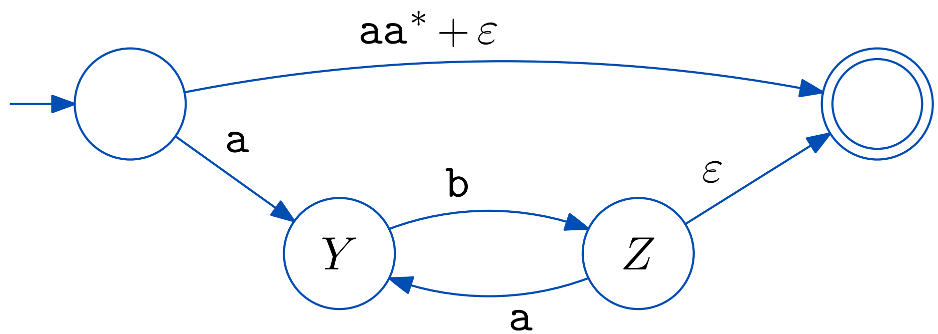
Example NFA

Here is the earlier NFA for $\mathtt{a}^* + (\mathtt{ab})^*$ adjusted to have a unique accept state.



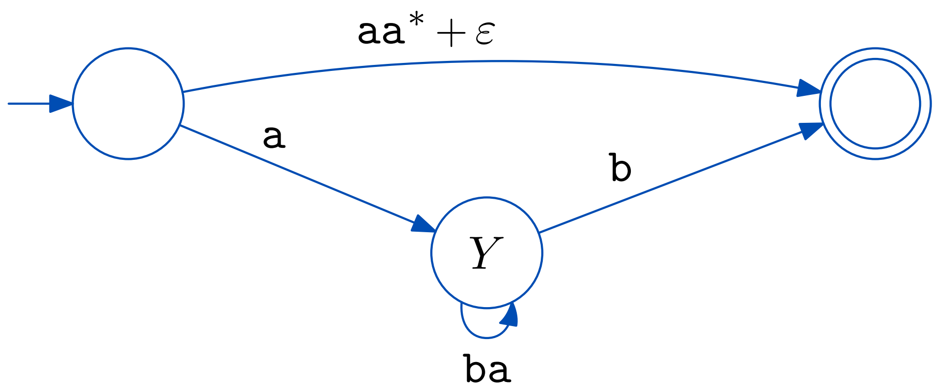
First Step

If we eliminate state X we get

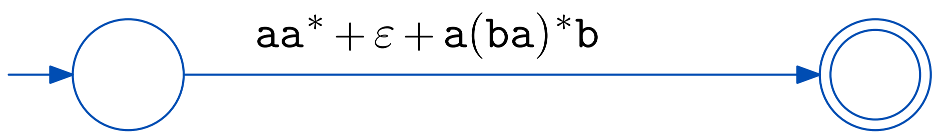


And Then

If we eliminate state Z we get

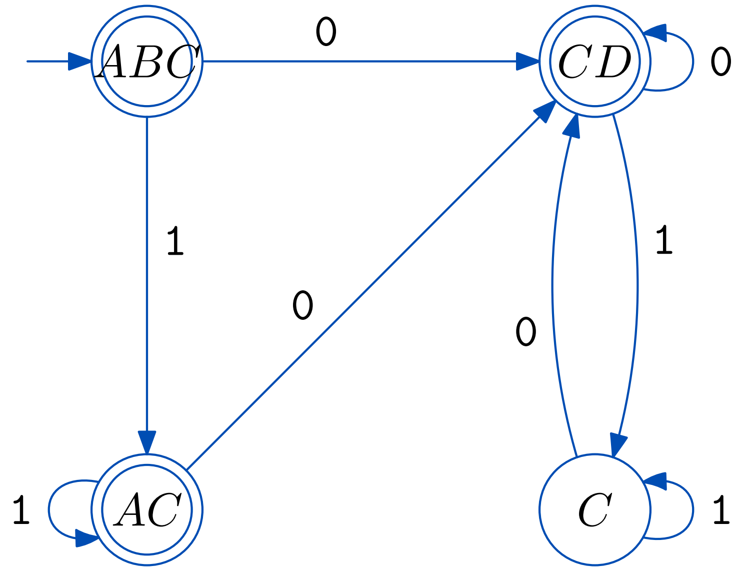


If we eliminate state Y we get



Practice

Convert that following DFA (from earlier) to an RE using the GFA method.



Solution to Practice

$$(0+11^*0)(0+11^*0)^*+\varepsilon+11^*$$

Summary

Kleene's theorem says that the following are equivalent for a language: there is an FA for it; there is an NFA for it; and there is an RE for it. The proof provides an algorithm to convert from one form to another; the conversion from NFA to DFA is the subset construction.