DFA to RE Conversion

Atonu Roy Chowdhury,
ext.atonu.roy@bracu.ac.bd

July 14, 2025

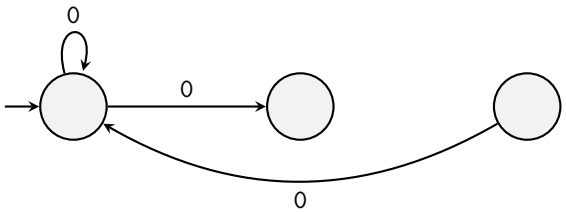
1 The Procedure

In order to convert a Deterministic Finite Automata (DFA) to a Regular Expression (RE), we have the following steps:

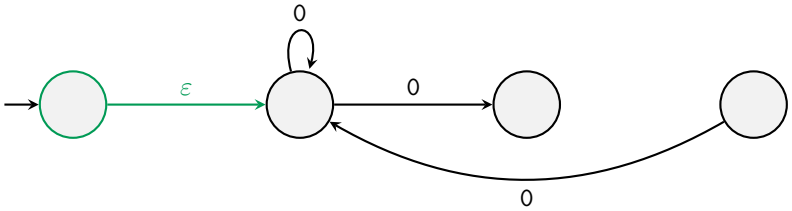
1. Preprocessing:

(a) The start state cannot have any incoming arrows. If there are any incoming arrows to the starting state, we introduce a new starting state, and add an epsilon transition from the new starting state to the old starting state.

For example, consider the following:

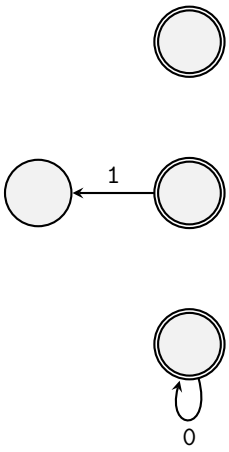


Here, the starting state has 2 incoming arrows. So we need to add a new starting state.

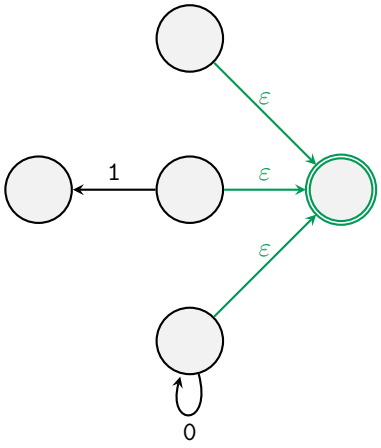


(b) There cannot be multiple accepting states. Moreover, the acccepting state cannot have outgoing arrows. The fix for these two is the same. We introduce a new accepting state, and add ε-transitions from the old accepting state(s) to the new accepting state, and remove the accepting status of the old accepting states.

For example, consider the following:

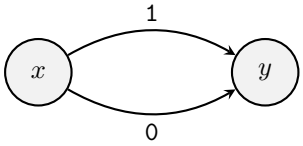


Here, there are outgoing arrows from accepting states. Furthermore, we have multiple accepting states. So we have to introduce a new accepting state.



(c) Given any two states x and y (it's allowed that $x = y$), there cannot be more than 1 arrow from x to y. If there are multiple arrows, we write the labels as union.

Consider this example:



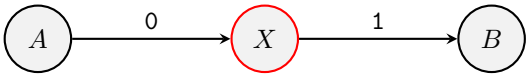
We shall write this as

2. State elimination: Once the preprocessing is complete, we pick any state (**NOT** the starting or the acccepting state) and eliminate it. By deleting a state, we make some "damage", so we need to "repair" the damage.

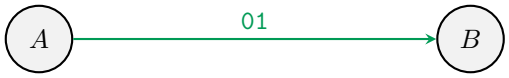
The order in which states are eliminated is not important for generating a regular expression. However, a different order may yield a different, albeit equivalent, regular expression.

The way you'd "repair the damage" is as follows:

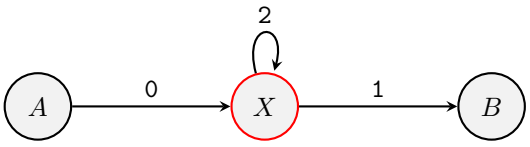
(a) Suppose you wanna eliminate the state X, which has an incoming edge from state A, and an outgoing edge to state B.



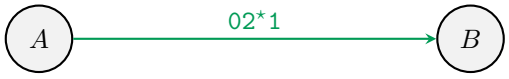
By deleting X, since we're losing the connection from A to B, we have to restore the connection. Since you'd go from A to B by reading `01`, the label of that arrow will be `01`.



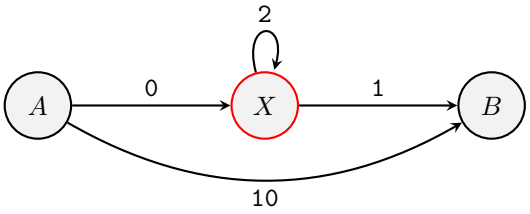
(b) Suppose you wanna eliminate the state X, which has an incoming edge from state A, and an outgoing edge to state B. Furthermore, X has a self-loop.



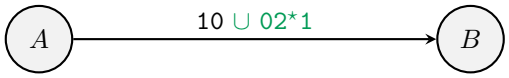
In this case, when we restore the connection from A to B, the label will be `02*1`.



(c) Suppose you wanna eliminate the state X, which has an incoming edge from state A, and an outgoing edge to state B. Furthermore, there's already a transition from A to B.



In this case, you can go from A to B either via `10` (already exists), or via `02*1`. So we label the arrow as the union of these two.

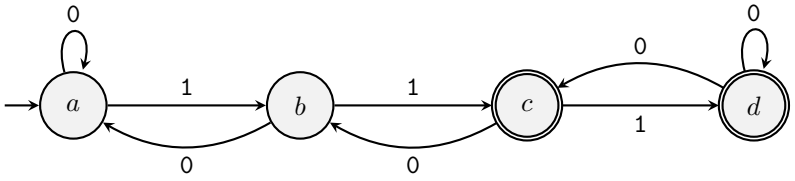


(d) If there's a trap state in the DFA, you can just remove it without doing anything else.

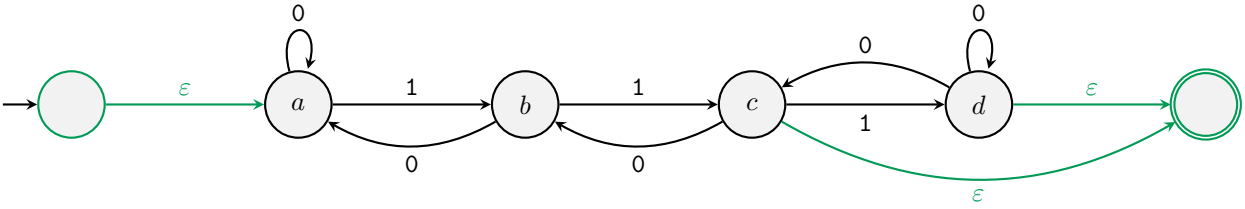
We keep on eliminating states until only two states (the starting state and the accepting state) remain. Then the label of the tranisiton from the starting state to the acccepting state will be the desired regular expression.

2 An Example

Let's now see an example to illustrate how this method works. Suppose we have the following DFA:



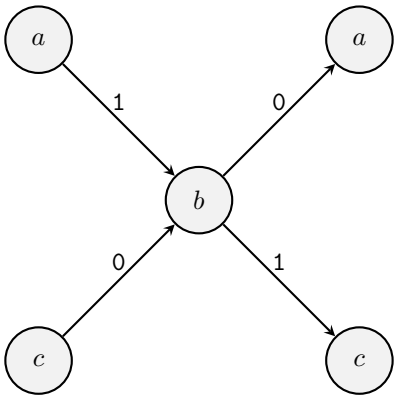
We have to do the preprocessing first. For that purpose, let's introduce a new starting state and a new accepting state.



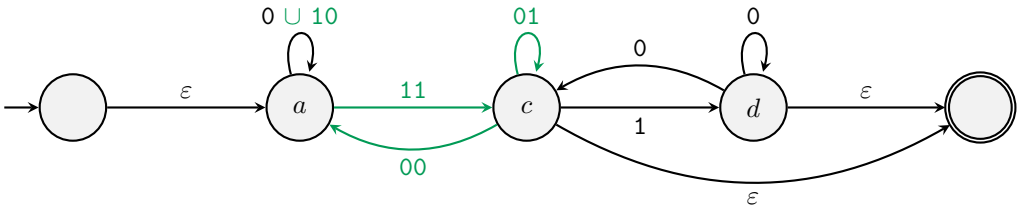
Let's eliminate b first. We have to keep track of the arrows incoming to b, and arrows outgoing from b. Then the number of arrows we need to repair is

$$\text{number of incoming arrows} \times \text{number of outgoing arrows}.$$

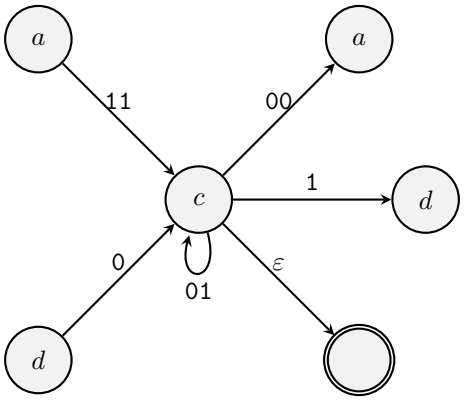
You may find the following "spider diagram" useful, to keep track of all the arrows incoming to b and outgoing from b.



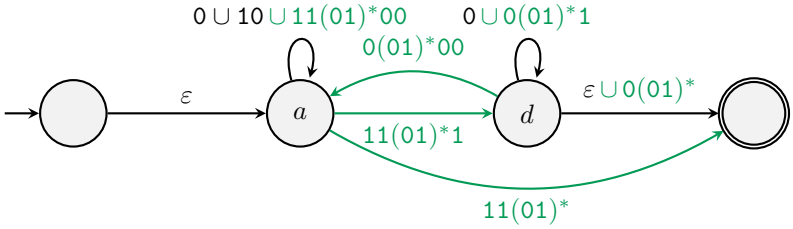
So we need to put `10`-transition from a to a; `11`-transition from a to c; `00`-transition from c to a; `01`-tranisiton from c to c.



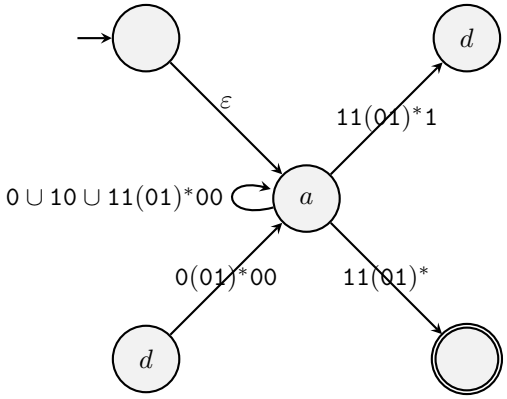
Next, let's eliminate c. The "spider diagram" for c looks like this:



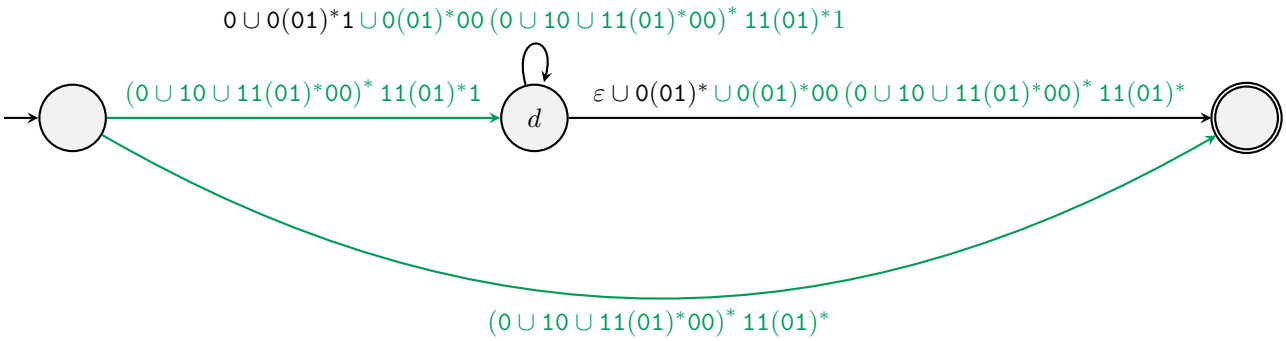
So we'll have $2 \times 3 = 6$ new transitions: $\mathtt{11(01)^*00}$ from a to a; $\mathtt{11(01)^*1}$ from a to d; $\mathtt{11(01)^*}$-transition from a to the acccepting state; $\mathtt{0(01)^*00}$ from d to a; $\mathtt{0(01)^*1}$ from d to d; $\mathtt{0(01)^*}$ from d to the acccepting state.



If we now eliminate a, the "spider diagram" for a will be



So there will be $2 \times 2 = 4$ new transitions: $\mathtt{(0 \cup 10 \cup 11(01)^*00)^*\ 11(01)^*}$ from the starting state to the accepting state; $\mathtt{(0 \cup 10 \cup 11(01)^*00)^*\ 11(01)^*1}$ from the starting state to d; $\mathtt{0(01)^*00\ (0 \cup 10 \cup 11(01)^*00)^*\ 11(01)^*1}$ from d to d; $\mathtt{0(01)^*00\ (0 \cup 10 \cup 11(01)^*00)^*\ 11(01)^*}$ from d to the acccepting state.



Finally, after eliminating d, we'll get