How to Transform a Finite Automaton into a Regular Expression

Formulation of the problem. We have a finite automaton. We want to describe a regular expression that covers exactly all the words acceted by this automaton and only these words.

Algorithm: first step.

- We add a new starting state, and
- we add a jump from the new starting state to the old starting state.

Algorithm: second step.

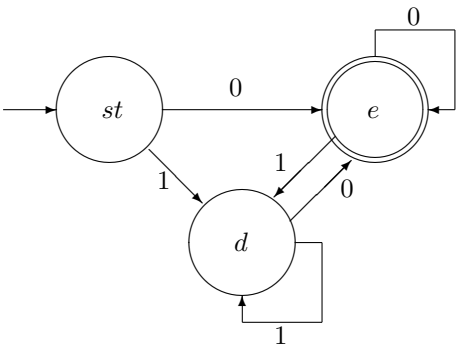
- We add a new final state;
- old final states are no longer final, and
- we add a jump from each old final state to the new final state.

Algorithm: following steps. We want to create an equivalent automation with only two states: starting state and final state. To do this, we eliminate all other states one by one.

When we eliminate state k, then for every other two states i and j, the expression $R_{i,j}$ that was previously written at the arrow from i to j is replaced by the new expression

$$R'_{i,j} = R_{i,j} \cup (R_{i,k} R^*_{k,k} R_{k,j}).$$

What is $R_{i,j}$: example. Let us illustrate it on the example of the following automaton that recognizes even unsigned binary integers, i.e., binary integers that end in 0. This automaton has three states: start st, even e, and odd d. (Please note that here, e *does not* mean error.)



Here:

- at the arrow going from $i = st$ to $j = e$, we have the symbol 0; this means that the only way to directly go from st to e is to read the symbol 0; so, the set of all the words that lead directly from st to e is

$$R_{st,e} = 0;$$

- there is no arrow going from $i = e$ to $j = st$, which means nothing will lead us directly from e to st; so, the state of all the words that lead directly from e to st is empty:

$$R_{e,st} = \emptyset.$$

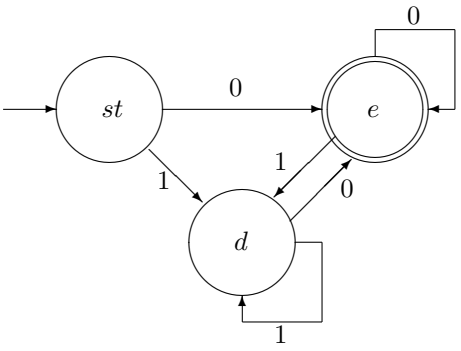
If we have two symbols a and b both placed at the arrow going from a state q to the state q', then to go directly from q to q', we would need to see either a or b. So, the set of all the words that lead directly from q to q' consists of two words: a and b. This set $\{a, b\}$ can be described, in terms of the regular expressions, as $a \cup b$. So, here, we write

$$R_{q,q'} = a \cup b.$$

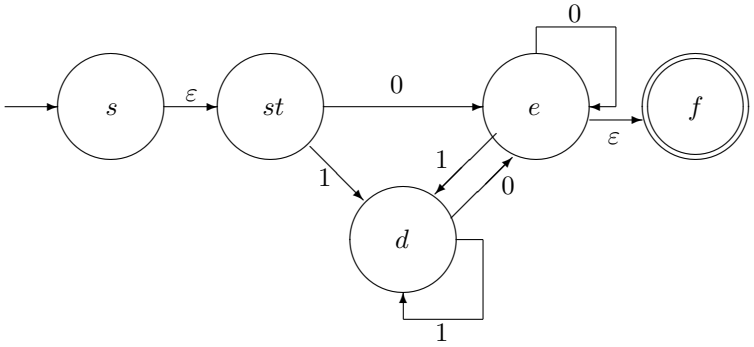
Motivation for the formula $R'_{i,j} = R_{i,j} \cup (R_{i,k} R^*_{k,k} R_{k,j})$. To get from i to j, we can:

- either go directly – this corresponds to $R_{i,j}$,
- or we:
  - first go to k (which corresponds to $R_{i,k}$),
  - then maybe make a few loops that bring us back to k (this corresponds to $R^*_{k,k}$), and
  - finally, go to j (this corresponds to $R_{k,j}$).

Example. Let us illustrate this procedure on the example of the following automaton that recognizes even unsigned binary integers, i.e., binary integers that end in 0. This automaton has three states: start st, even e, and odd d. (Please note that here, e *does not* mean error.)



First two steps. First, we add a new starting state s, a new final state f, and the two jumps:



Eliminating the state st. Here, in addition to the states s and f, we have three other states: st, e, and d. Let us start eliminating the states one by one.

It does not matter what state we start with, but, interestingly, the resulting regular expressions may be different depending on what state we start with. So, if you and your friend get two different expressions, this does not mean that one of these expressions is wrong: both may be correct expressions for the same language.

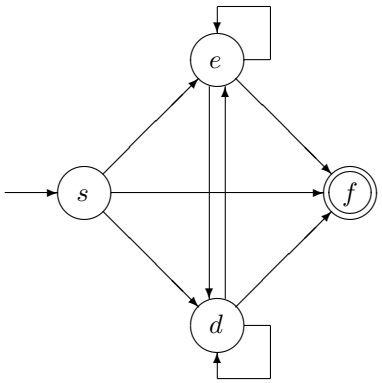
Let us start left-to-right, with eliminating the state st. Once we eliminate it, we will have an automaton with 4 states: s, f, e, and d.

First thing we do, we draw all possible arrows connecting these states, with two exceptions:

- from the start state, you can only go out, and

- into the final state, you can only go in.

Thus, we have the following picture:



For each arrow connecting the vertices i and j, we need to compute the new expression R'_{ij} by using the general formula

$$R'_{i,j} = R_{i,j} \cup (R_{i,k} R^*_{k,k} R_{k,j}).$$

In this formula, k is the state that is being eliminated. In our case, we eliminate the state st, so $k = st$, and thus, the formula takes the form

$$R'_{i,j} = R_{i,j} \cup (R_{i,st} R^*_{st,st} R_{st,j}).$$

Let us first compute the expressions $R'_{i,j}$ for the arrows that start at the first left-to-right state s. We need to compute the expressions $R'_{s,e}$, $R'_{s,d}$, and $R'_{s,f}$. Let us compute them one by one, top to bottom, starting from e and ending in d.

Computing $R'_{s,e}$. Here, $i = s$, $j = e$, so the general formula takes the form

$$R'_{s,e} = R_{s,e} \cup (R_{s,st} R^*_{st,st} R_{st,e}).$$

Here, in the original automaton (with 5 states), there is no arrow going from s to e, so $R_{s,e}$ is the empty set:

$$R_{s,e} = \emptyset.$$

From s to st there is an arrow with an empty string symbol ε on top, so $R_{s,st}$ is the empty string:

$$R_{s,st} = \Lambda.$$

In the 5-state graph, there is no arrow going from st to st, so $R_{st,st} = \emptyset$. The Kleene star of each language consists of the empty string and of elements from

this set repeated several times. The empty set has no elements, so its Kleene star is simply the empty string:

$$\emptyset^* = \Lambda,$$

and thus:

$$R^*_{st,st} = \emptyset^* = \Lambda.$$

Finally, in the 5-state graph, at the arrow going from st to e, we have the symbol 0, so

$$R_{st,e} = 0.$$

Substituting these expressions into the above formula, we conclude that

$$R'_{s,e} = \emptyset \cup (\Lambda\Lambda 0).$$

If we concatenate any string with the empty string, this does not change the original string. So, in general, for every language A, we have

$$\Lambda A = A\Lambda = A.$$

In our case, $\Lambda\Lambda 0 = 0$, so we conclude that

$$R'_{s,e} = \emptyset \cup 0.$$

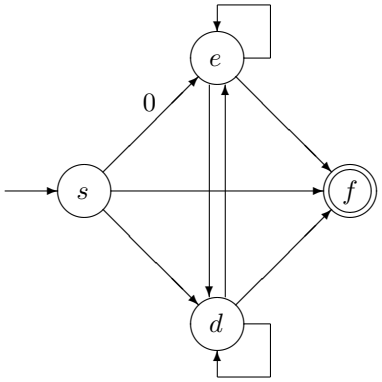
Taking a union of the empty set and any set means that we do not add anything to the original set. So, in general, for every language A, we have

$$\emptyset \cup A = A \cup \emptyset = A.$$

In particular, in our case, we have

$$R'_{s,e} = 0.$$

So, in the resulting 4-state automaton, we place 0 at the arrow that leads from s to e:



Computing $R'_{s,f}$. Here, $i = s$, $j = f$, k is still st – the state that we are eliminating, so the general formula has the form

$$R'_{s,f} = R_{s,f} \cup (R_{s,st} R^*_{st,st} R_{st,f}).$$

Here, in the 5-state graph, there are no arrows going from s to f, so

$$R_{s,f} = \emptyset.$$

We already know that $R_{s,st} = \Lambda$ and that $R^*_{st,st} = \Lambda$.

In the 5-state graph, there is also no arrow going from st to f, so

$$R_{st,f} = \emptyset.$$

The whole formula takes the form

$$R'_{s,f} = \emptyset \cup (\Lambda\Lambda\emptyset).$$

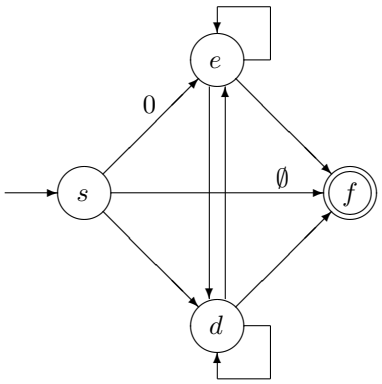
We already know that concatenation with the empty string Λ does not change the language, so $\Lambda\Lambda\emptyset = \emptyset$, and the formula takes the form

$$R'_{s,f} = \emptyset \cup \emptyset.$$

The union of two empty sets is empty: if we had two sets with no elements in them, and we put them together, there are still no elements. So, here

$$R'_{s,f} = \emptyset.$$

In the 4-element graph, we place this expression at the arrow that goes from s to f:



Computing $R'_{s,d}$. From the state s, the only remaining state is the state d.

Here, $i = s$, $j = d$, k is still st – the state that we are eliminating, so the general formula has the form

$$R'_{s,d} = R_{s,d} \cup (R_{s,st} R^*_{st,st} R_{st,d}).$$

Here, in the 5-state graph, there are no arrows going from s to d, so

$$R_{s,d} = \emptyset.$$

We already know that $R_{s,st} = \Lambda$ and that $R^*_{st,st} = \Lambda$.

In the 5-state graph, at the arrow going from st to d, we have 1, so

$$R_{st,d} = 1.$$

The whole formula takes the form

$$R'_{s,d} = \emptyset \cup (\Lambda\Lambda 1).$$

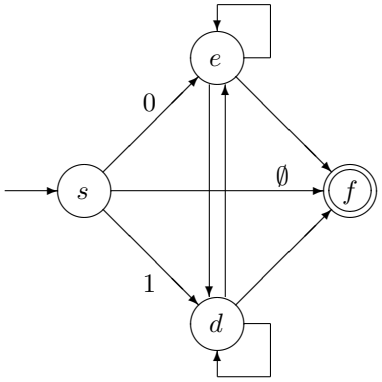
We already know that concatenation with the empty string Λ does not change the language, so $\Lambda\Lambda 1 = 1$, and the formula takes the form

$$R'_{s,d} = \emptyset \cup 1.$$

We also know that the union with the empty set does not change the original set, so here:

$$R'_{s,d} = 1.$$

In the 4-element graph, we place this expression at the arrow that goes from s to d:



Computing $R'_{e,e}$. Let us now compute the expressions for which the starting state is e.

We start with the expression $R'_{e,e}$, with $i = j = e$ and $k = st$, which has the form:

$$R'_{e,e} = R_{e,e} \cup (R_{e,st} R^*_{st,st} R_{st,e}).$$

In the 5-state graph, at the arrow doing from e to e we have 0, so

$$R_{e,e} = 0.$$

In the 5-state graph, there is no arrow going from e to st, so

$$R_{e,st} = \emptyset.$$

Concatenation of the empty set with any other set A means we first take the word from the empty set, and then the word from the set A. Since the empty set has no elements, this means that

$$\emptyset A = A\emptyset = \emptyset.$$

In our case,

$$R_{e,st} R^*_{st,st} R_{st,e} = \emptyset R^*_{st,st} R_{st,e} = \emptyset,$$

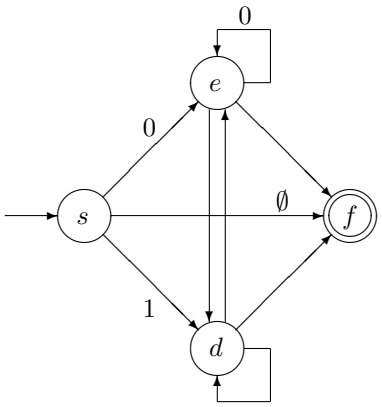
so

$$R'_{e,e} = 0 \cup \emptyset.$$

The union of any set with the empty set does not change anything, so

$$R'_{e,e} = 0.$$

In the 4-element graph, we place this expression at the arrow that goes from e to e:



Computing $R'_{e,d}$. In this case, $i = e$, $j = d$, and $k = st$, so

$$R'_{e,d} = R_{e,d} \cup (R_{e,st} R^*_{st,st} R_{st,d}).$$

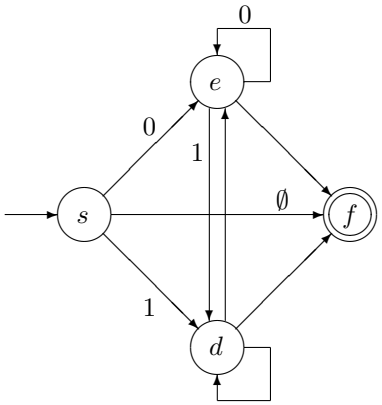
In the 5-element graph, we have 1 at the arrow that goes from e to d, so

$$R_{e,d} = 1.$$

We already know that $R_{e,st} = \emptyset$, thus $R_{e,st}R^*_{st,st}R_{st,d} = \emptyset$, and

$$R'_{e,d} = 1 \cup \emptyset = 1.$$

In the 4-element graph, we place this expression at the arrow that goes from e to d:



Computing $R'_{e,f}$. Here, $i = e$, $j = f$, and $k = st$, so

$$R'_{e,f} = R_{e,f} \cup (R_{e,st}R^*_{st,st}R_{st,f}).$$

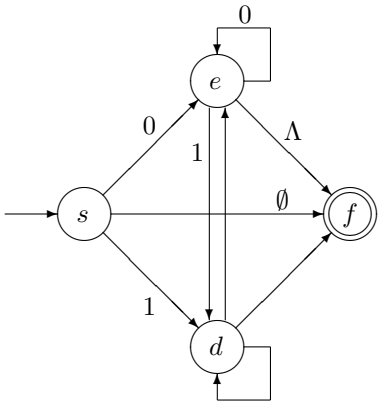
In the 5-element graph, at the arrow that goes from e to f, we have the symbol ε that indicates the empty string, so

$$R_{e,f} = \Lambda.$$

We already know that $R_{e,st} = \emptyset$, thus $R_{e,st}R^*_{st,st}R_{st,f} = \emptyset$, and

$$R'_{e,f} = \Lambda \cup \emptyset = \Lambda.$$

In the 4-element graph, we place this expression at the arrow that goes from e to f:



Computing $R'_{d,d}$. Finally, we need to deal with arrows coming out of the state d.

Let us start with $R'_{d,d}$. Here, $i = j = d$ and $k = st$, so

$$R'_{d,d} = R_{d,d} \cup (R_{d,st} R^*_{st,st} R_{st,d}).$$

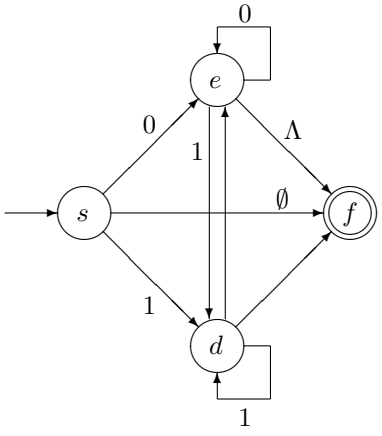
In the 5-element graph, at the arrow that goes from d to d, we have 1, so

$$R_{d,d} = 1.$$

In the 5-element graph, there is no arrow going from d to st, so $R_{d,st} = \emptyset$ and thus $R_{d,st} R^*_{st,st} R_{st,d} = \emptyset$, and

$$R'_{d,d} = 1 \cup \emptyset = 1.$$

In the 4-element graph, we place this expression at the arrow that goes from d to d:



Computing $R'_{d,e}$. Here, $i = d$, $j = e$, and $k = st$, so

$$R'_{d,e} = R_{d,e} \cup (R_{d,st} R^*_{st,st} R_{st,e}).$$

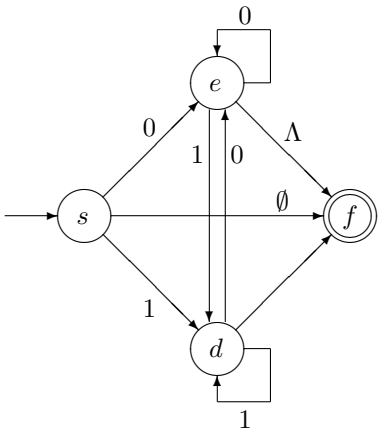
In the 5-element graph, at the going from d to e, we have 0, so

$$R_{d,e} = 0.$$

We already know that $R_{d,st} = \emptyset$ and thus $R_{d,st} R^*_{st,st} R_{st,f} = \emptyset$, and

$$R'_{d,e} = 0 \cup \emptyset = 0.$$

In the 4-element graph, we place this expression at the arrow that goes from d to e:



Computing $R'_{d,f}$. The only remaining expression is $R'_{d,f}$. Here, $i = d$, $j = f$, and $k = st$, so

$$R'_{d,f} = R_{d,f} \cup (R_{d,st} R^*_{st,st} R_{st,f}).$$

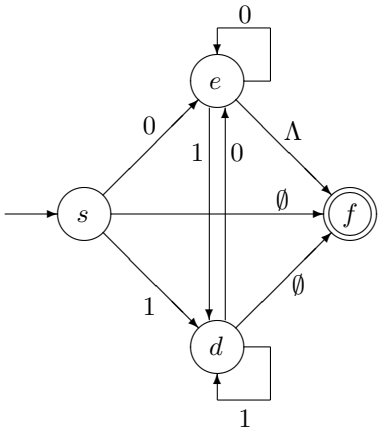
In the 5-element graph, there are no arrows going from d to f, so

$$R_{d,f} = \emptyset.$$

We already know that $R_{d,st} = \emptyset$ and thus $R_{d,st} R^*_{st,st} R_{st,f} = \emptyset$, and

$$R'_{d,f} = \emptyset \cup \emptyset = \emptyset.$$

In the 4-element graph, we place this expression at the arrow that goes from d to f:



We have eliminated the state st. Now, we finally have a 4-element graph from which the state st has been eliminated. Now, we need to start with this new graph and eliminate one of the remaining two states: e and d. Let us start with eliminating the state e.

General rules. Before we do that, let us summarize the general rules that were helpful so far:

$$A\emptyset = \emptyset A = \emptyset;$$

$$\emptyset^* = \Lambda;$$

$$\Lambda A = A\Lambda = A;$$

$$A \cup \emptyset = \emptyset \cup A = A.$$

Important comment. It is important to distinguish between an empty string and an empty set. Let me give a simple example.

- If your set of passwords consists of an empty string Λ, this means that once you hit Enter, you are in.
- On the other hand, if your set of passwords is an empty set $\emptyset$, this means that there is no password at all, no matter what you type, you will not get in.

The difference is drastic when you consider if you consider concatenation AB of two languages A and B, i.e., the set of all the two-part words ab, where part a is taken from the language A, and part b is taken from the language B.

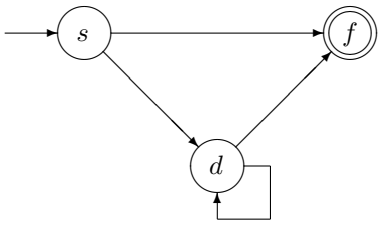
- If B is the empty string $B = \Lambda$, then writing first a word a and then an empty string — i.e., nothing more – results in the same word a, so $A\Lambda = A$.

- On the other hand, if B is the empty set $B = \emptyset$, then, since the empty set has no elements b at all, we cannot have a two-part word of the type ab, so $A\emptyset = \emptyset$.

Designing a 3-element automaton. We want to have an automaton with 3 states: s, f, and d. Let us draw all possible arrows, with two exceptions:

- from the start state, you can only go out, and
- into the final state, you can only go in.

Thus, we have the following picture:



Computing the values R'_{ij} for the 3-state automaton. For $i = s$, $j = f$, and $k = e$, we have

$$R'_{s,f} = R_{s,f} \cup (R_{s,e} R^*_{e,e} R_{e,f}).$$

Substituting the values R_{ij} from the above 4-state automaton, we get

$$R'_{s,f} = \emptyset \cup (00^* \Lambda).$$

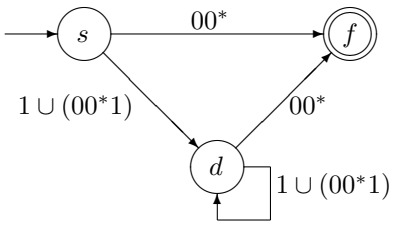
By using the general rules, we conclude that

$$R'_{s,f} = 00^*.$$

Similarly,

$$R'_{s,d} = R_{s,d} \cup (R_{s,e} R^*_{e,e} R_{e,d}) = 1 \cup (00^*1);$$
$$R'_{d,d} = R_{d,d} \cup (R_{d,e} R^*_{e,e} R_{e,d}) = 1 \cup (00^*1);$$
$$R'_{d,f} = R_{d,f} \cup (R_{d,e} R^*_{e,e} R_{e,f}) = \emptyset \cup (00^* \Lambda) = 00^*.$$

Thus, the resulting 3-state automaton has the following form:

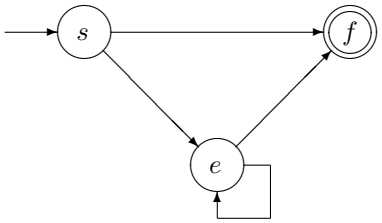


Towards the final answer. Now all we need to do is eliminate the remaining state d. The resulting expression for $R'_{s,f}$ is the desired regular expression corresponding to the original automaton:

$$R'_{s,f} = R_{s,f} \cup (R_{s,d} R^*_{d,d} R_{d,f}) =$$
$$00^* \cup ((1 \cup (00^*1))(1 \cup (00^*1))^* 00^*).$$

What if, based on the 4-state automaton, we first eliminate the state d? In the above text, once we reached the 4-state configuration with two extra states e and d, we first eliminated the state e. What if, instead we first eliminate the state d?

Then, we will get the following picture:



For $i = s$, $j = f$, and $k = d$, we have

$$R'_{s,f} = R_{s,f} \cup (R_{s,d} R^*_{d,d} R_{d,f}) = \emptyset \cup (11^*\emptyset),$$

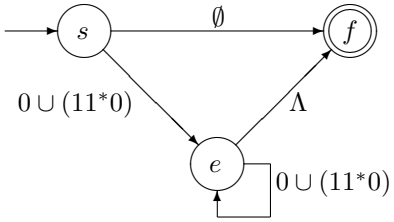
so

$$R'_{s,f} = \emptyset.$$

Similarly,

$$R'_{s,e} = R_{s,e} \cup (R_{s,d} R^*_{d,d} R_{d,e}) = 0 \cup (11^*0);$$
$$R'_{e,e} = R_{e,e} \cup (R_{e,d} R^*_{d,d} R_{d,e}) = 0 \cup (11^*0);$$
$$R'_{e,f} = R_{e,f} \cup (R_{e,d} R^*_{d,d} R_{d,f}) = \Lambda \cup (11^*\emptyset) = \Lambda.$$

Thus, the resulting 3-state automaton has the following form:



Now all we need to do is eliminate the remaining state e. The resulting expression for $R'_{s,f}$ is the desired regular expression corresponding to the original automaton:

$$R'_{s,f} = R_{s,f} \cup (R_{s,e} R^*_{e,e} R_{e,f}) =$$
$$\emptyset \cup ((0 \cup (11^*0))(0 \cup (11^*0))^* \Lambda) =$$
$$(0 \cup (11^*0))(0 \cup (11^*0))^*.$$

As we warned, this is a *different* regular expression than what we got when we first eliminated the state e – but the resulting language is the same, it is the set of all binary sequences that end in 0.

Practice. Apply the general algorithm to some other automaton.

Important comments.

- If in the original automaton we have two or more different symbols a, b, ... leading from state q to state q', this means that using any of these symbols leads us to the new state, i.e., that the set of all states leading from q to q' is the *union* $a \cup b \cup \ldots$.

- If the original automaton has non-final sink states, then they can be eliminating by simply deleting them.

  Indeed, eliminating the sink state e means that for all other states i and j, we get $R'_{i,j} = R_{i,j} \cup (R_{i,e} R^*_{e,e} R_{e,j})$. By definition of a sink state, it has no arrows going from it to any other states. Thus, we always have $R_{e,j} = \emptyset$. Concatenation with the empty set $R^*_{e,j}$ is empty set, so we always have $R_{i,e} R^*_{e,e} R_{e,j} = \emptyset$. Union of any set with the empty set is that same original set, so we have $R'_{i,j} = R_{i,j}$. Thus indeed, eliminating the sink state does not change any other transitions.