Converting regular expressions to state diagrams and vice versa

This material is from Chapter 9 in the textbook.

We show that regular expressions and state diagrams define exactly the same class of languages. This fact is remarkable because superficially state diagrams and regular expressions appear to be quite different.

Regular expressions to state diagrams

For an arbitrary regular expression we construct a state diagram with ε-transitions that satisfies the following conditions:

1. There is exactly one accepting state and it is distinct from the starting state.
2. There are no transitions into the starting state.
3. There are no outgoing transitions from the accepting state.

The recursive construction relies on the fact that the previously constructed state-transition diagrams satisfy the above conditions 1., 2., 3.

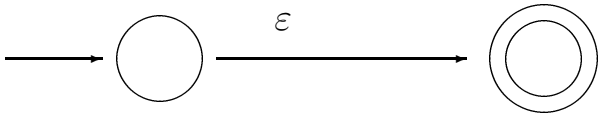
The state diagrams for the base cases (i) $E = \emptyset$, (ii) $E = \varepsilon$, (iii) $E = a$, $(a \in \Sigma)$ are given in Figure 1.

Next suppose that E_i, $i = 1, 2$, are regular expressions and we have constructed a state diagram S_i that is equivalent to E_i and satisfies the conditions 1., 2., 3., see Figure 2.

We have to show how to construct state diagrams for the regular expressions $E_1 + E_2$, $E_1 \cdot E_2$ and E_1^*.

(i) $E = \emptyset$

(ii) $E = \varepsilon$



(iii) $E = a$

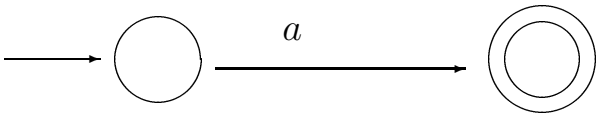


Figure 1: State diagrams for the base cases.

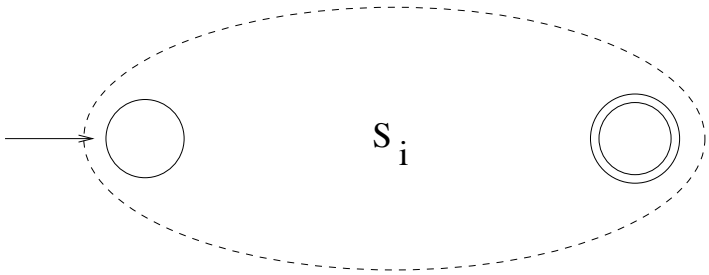


Figure 2: State diagram S_i that is constructed for the regular expression E_i, $i = 1, 2$.

(iv) The state diagram for $E_1 + E_2$ is depicted in Figure 3.

(v) The state diagram for the regular expression $E_1 \cdot E_2$ is constructed by merging the accepting state of S_1 with the starting state of S_2, the merged state is not accepting. See Figure 4.

(vi) For the regular expression E_1^* we construct a state diagram as follows: merge the accepting state of S_1 with the starting state of S_1, the merged state is neither an accepting nor the starting state. Then we introduce a new starting state and a new

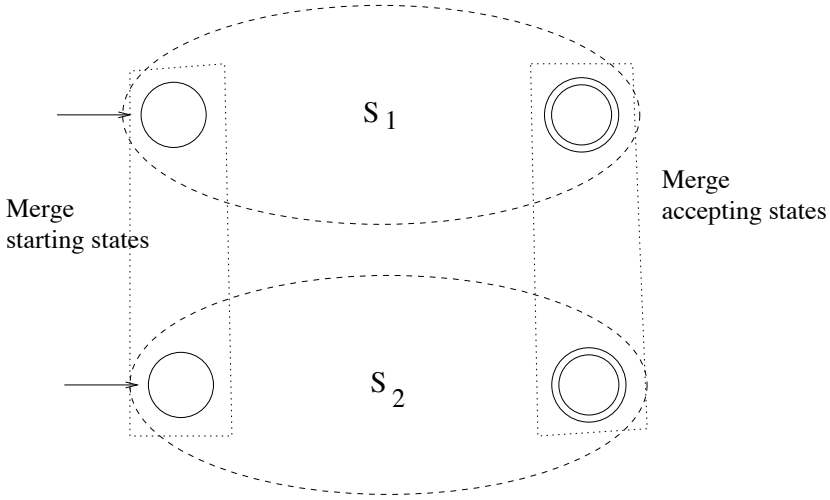


Figure 3: State diagram for regular expression $E_1 + E_2$.

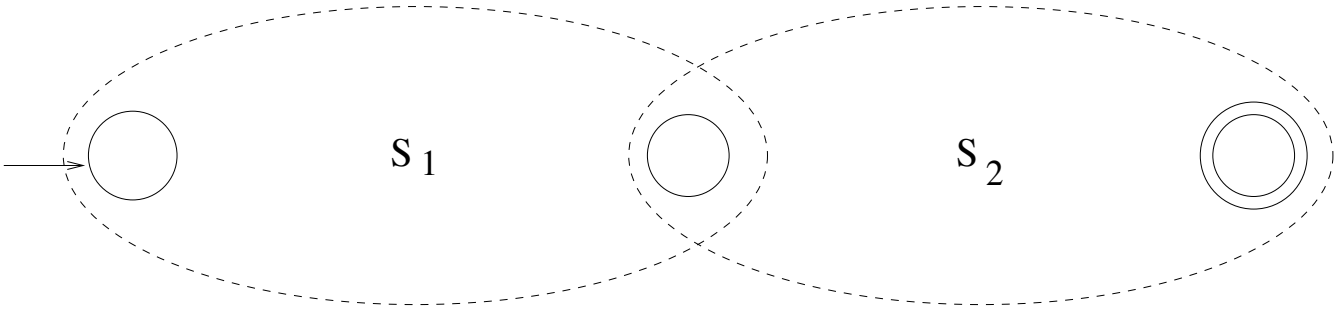


Figure 4: State diagram for the regular expression $E_1 \cdot E_2$.

accepting state as depicted in Figure 5.

It should be noted that in each of the above cases (iv), (v) and (vi) the resulting state diagram again satisfies the properties 1., 2., 3.

Using techniques from the previous chapter (last week) we can eliminate ε-transitions and convert the state diagram into an equivalent deterministic state diagram.

Example. Apply the construction to the regular expression $(01 + 1)^*1$.

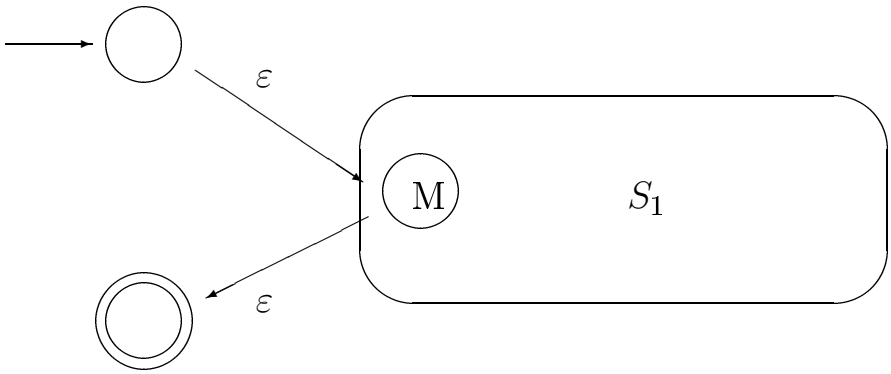


Figure 5: State diagram for the regular expression E_1^*.

State diagrams to regular expressions (Section 9.2)

For a given state diagram we construct an equivalent regular expression using the so called state-elimination procedure. The intermediate stages in the construction are *generalized state-transition diagrams* where the edges can be labeled by any regular expressions (instead of individual alphabet symbols).

The construction is given on pp. 196–197 in the textbook and we consider here an example.

Example. We consider the state diagram given in Figure 6.

In the first stage of the construction we modify the state diagram so that it has only one accepting state. The resulting state diagram is given in Figure 7, in the figure e stands for the empty string.

Assuming we apply the state elimination technique to the state diagram of Figure 7 by first eliminating the state r and then the state s, the resulting regular expression

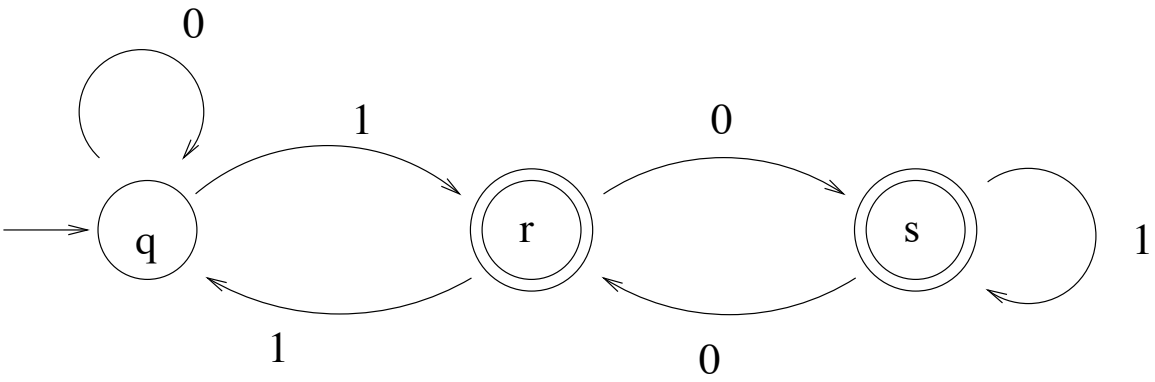


Figure 6: A state diagram

will be

$$(0 + 11 + (10(1 + 00)^*01))^*(1 + 10(1 + 00)^*(0 + \varepsilon))$$

The details will be done in class.

Note: When using the state-elimination procedure, we can eliminate the states in an arbitrary order (the starting and the accepting state are not eliminated). Using different orders may (and usually will) produce different regular expressions, however they all denote the same language as the original state diagram.

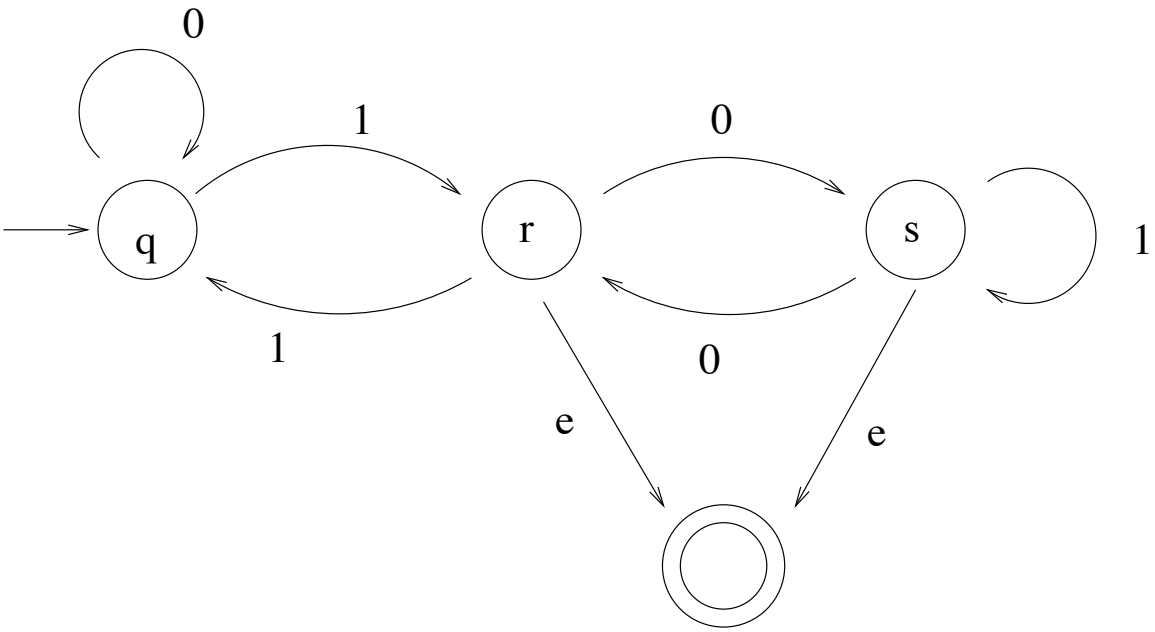


Figure 7: The modified state diagram that has only one accepting state. In the figure e denotes the empty string.