

# CSCI565 - Compiler Design

Spring 2010

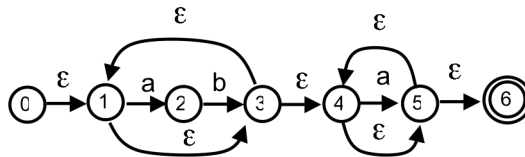
## Homework 1 - Solution

**Problem 1 [40 points]:** Consider the alphabet  $\Sigma = \{a,b\}$ .

- Construct a Non-Deterministic-Finite Automaton (NFA) using the Thompson construction that is able to recognize the sentences generated by the regular expression RE =  $(ab)^* \cdot (a)^*$ .
- Do the sentences  $w_1 = \text{“abaa”}$  and  $w_2 = \text{“aaa”}$  belong to the language generated by this regular expression? Justify.
- Convert the NFA in part a) to a DFA using the subset construction. Show the mapping between the states in the NFA and the resulting DFA.
- Minimize the DFA using the iterative refinement algorithm discussed in class. Show your intermediate partition results and double check the DFA using the sentences  $w_1$  and  $w_2$ .

**Solution:**

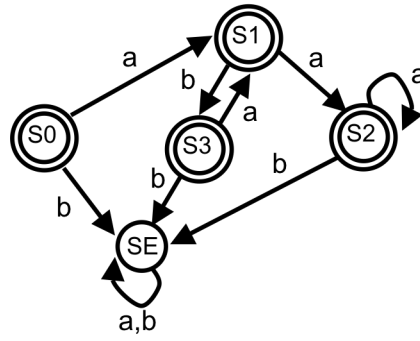
- A possible construction (already simplified to have only a single  $\epsilon$ -transition between states) would result in the NFA shown below and where the start state is labeled 0.



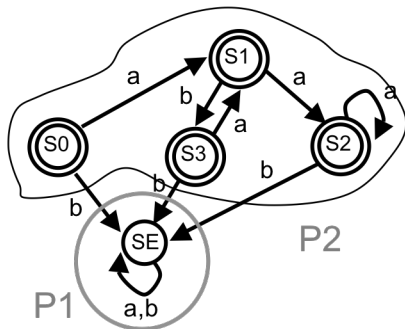
- Both words are recognized by this NFA. Regarding the word “abaa” there is a path from state 0 to the accepting state 6, namely: 0,1,2,3,4,5,4,5,6. Regarding the word “aaa” the automaton may reach state 6 following the path 0,1,3,4,5,4,5,4,5,6.
- Using the subset construction we arrive at the following subsets and transitions.

$$\begin{aligned}
 S_0 &= \epsilon\text{-closure}(0) &= \{0, 1, 3, 4, 5, 6\} &\text{– this is a final state because of state 6} \\
 S_1 &= \text{DFAedge}(S_0, a) &= \epsilon\text{-closure}(\text{goto}(S_0, a)) &= \{2, 4, 5, 6\} &\text{– final state} \\
 S_E &= \text{DFAedge}(S_0, b) &= \epsilon\text{-closure}(\text{goto}(S_0, b)) &= \{\} \\
 S_2 &= \text{DFAedge}(S_1, a) &= \epsilon\text{-closure}(\text{goto}(S_1, a)) &= \{4, 5, 6\} &\text{– final state} \\
 S_3 &= \text{DFAedge}(S_1, b) &= \epsilon\text{-closure}(\text{goto}(S_1, b)) &= \{3, 4, 5, 6\} &\text{– final state} \\
 \\ 
 \text{DFAedge}(S_2, a) &= \epsilon\text{-closure}(\text{goto}(S_2, a)) &= \{4, 5, 6\} &= S_2 \\
 \text{DFAedge}(S_2, b) &= \epsilon\text{-closure}(\text{goto}(S_2, b)) &= \{\} &= S_E \\
 \text{DFAedge}(S_3, a) &= \epsilon\text{-closure}(\text{goto}(S_3, a)) &= \{2, 4, 5, 6\} &= S_1 \\
 \text{DFAedge}(S_3, b) &= \epsilon\text{-closure}(\text{goto}(S_3, b)) &= \{\} &= S_E
 \end{aligned}$$

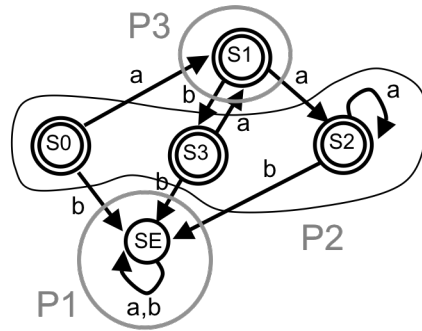
This results in the DFA shown below with starting state  $S_0$ .



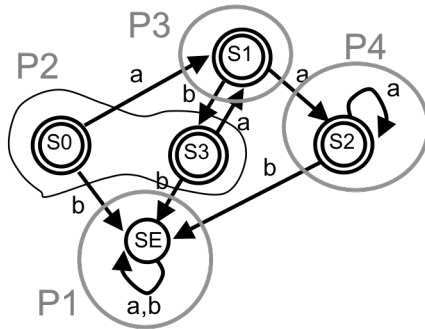
d) This DFA can be further minimize by using the iterative refinement partitioning yielding the sequence of partitions indicated below.



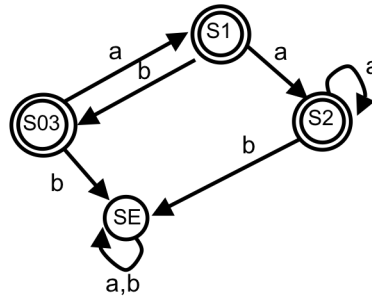
Initial partitioning on final states



Partitioning P2 on b

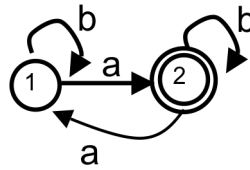


Partitioning P2 on a



final minimal DFA

**Problem2 [30 points]:** Consider the DFA below with starting state 1 and accepting state 2:



- Describe in English the set of strings accepted by this DFA.
- Using the Kleene construction algorithm derive the regular expression recognized by this automaton simplifying as much as possible.

**Solution:**

- This automaton recognizes all the strings over the  $\{a,b\}$  alphabet that are not empty either are the singleton “a”, or end with an “b” and can have any combination of pairs of “aa” sequences or a sequence of “b”s of arbitrary length.
- The derivations are shown below with the obvious simplification.

Expressions for  $k = 0$

$$R_{11}^0 = (b|\epsilon)$$

$$R_{12}^0 = (a)$$

$$R_{21}^0 = (a)$$

$$R_{22}^0 = (b|\epsilon)$$

Expressions for  $k = 1$

$$R_{11}^1 = R_{11}^0 (R_{11}^0)^* R_{11}^0 \quad R_{11}^0 = (b|\epsilon) \cdot (b|\epsilon)^* \cdot (b|\epsilon) \mid (b|\epsilon) \quad = b^*$$

$$R_{12}^1 = R_{11}^0 (R_{11}^0)^* R_{12}^0 \quad R_{12}^0 = (b|\epsilon) \cdot (b|\epsilon)^* \cdot (a) \mid (a) \quad = (b|\epsilon)^* (a) = b^*a$$

$$R_{21}^1 = R_{21}^0 (R_{11}^0)^* R_{21}^0 \quad R_{21}^0 = (a) \cdot (b|\epsilon)^* \cdot (b|\epsilon) \mid (a) \quad = ab^*$$

$$R_{22}^1 = R_{21}^0 (R_{11}^0)^* R_{22}^0 \quad R_{22}^0 = (a) \cdot (b|\epsilon)^* \cdot (a) \mid (b|\epsilon) \quad = a(b^*a \mid \epsilon) \mid (a|\epsilon) = ab^*a \mid b \mid \epsilon$$

Expressions for  $k = 2$

$$R_{11}^2 = R_{12}^1 (R_{22}^1)^* R_{21}^1 \quad R_{11}^1 = (b^*a) \cdot (ab^*a \mid b \mid \epsilon)^* \cdot ab^* \mid b^*$$

$$R_{12}^2 = R_{12}^1 (R_{22}^1)^* R_{12}^1 \quad R_{12}^1 = (b^*a) \cdot (ab^*a \mid b \mid \epsilon)^* \cdot (ab^*a \mid b \mid \epsilon) \mid b^*a$$

$$R_{21}^2 = R_{12}^1 (R_{22}^1)^* R_{21}^1 \quad R_{21}^1 = (b^*a) \cdot (ab^*a \mid b \mid \epsilon)^* \cdot ab^* \mid ab^*$$

$$R_{22}^2 = R_{12}^1 (R_{22}^1)^* R_{22}^1 \quad R_{22}^1 = (b^*a) \cdot (ab^*a \mid b \mid \epsilon)^* \cdot (ab^*a \mid b \mid \epsilon) \mid (ab^*a \mid b \mid \epsilon)$$

$$L = R_{12}^2 = (b^*a) \cdot (ab^*a \mid b \mid \epsilon)^* \cdot (ab^*a \mid b \mid \epsilon) \mid b^*a$$

As can be seen the simplification of any of these regular expressions beyond the expressions for  $k=1$  is fairly complicated. This method, although correct by design leads to regular expressions that are far from being a minimal or most compact representation of the regular language a given finite automaton can recognize.

**Problem 3 [10 points]:** Given a regular language  $L$ . *i.e.*, a language described by a regular expression, prove that the reverse of  $L$  is also a regular language (**Note:** the reverse of a language  $L$  is  $L^R$  where for each word  $w$  in  $L$ ,  $w^R$  is in  $L^R$ . Given a word  $w$  over the given alphabet,  $w^R$  is constructed by spelling  $w$  backwards).

**Solution:** If  $L$  is a regular language then there exists a DFA  $M_1$  that recognizes it. Now given  $M_1$  we can construct  $M_2$  that recognizes the reverse of  $L$  with respect to the input alphabet. We now describe how to construct  $M_2$ .  $M_2$  is a replica of  $M_1$  but reversing all the edges. The final state of  $M_2$  is the state that used to be the start state of  $M_1$ . The start state of  $M_2$  is a new state with  $\epsilon$ -transitions to the states in  $M_2$  that used to be the final states of  $M_1$ . Now because  $M_1$  might have multiple final states  $M_2$  is by construction a NFA. Given the equivalence of NFA and regular expressions we have shown that if  $L$  is regular so is  $L^R$ .

**Problem 4 [20 points]:** Draw the DFA capable of recognizing the set of all strings beginning with a 1 which interpreted as the binary representation of an integer (assuming the last digit to be processed is the least significant) is congruent to zero modulo 3 *i.e.*, the numeric value of this binary representation is a multiple of 3.

**Solution:** The hard part about this problem is that you need to keep track with the already observed bits what the remainder of the division by 3 is. Given that you have a remainder you would need no more than 2 states, one for each of the remainder values 1 through 2 being the state that represents a remainder of zero the accepting state, in this case state  $S_3$ . The DFA below accomplishes this. You can verify this DFA by trying the number 12 in binary 1100 or 21 in binary 10101. Notice that in the last state  $S_3$  any additional 0 means you are shifting the bits by one bit, *i.e.*, multiplying by 2, hence staying in the same state.

