Compilers

Spring 2009

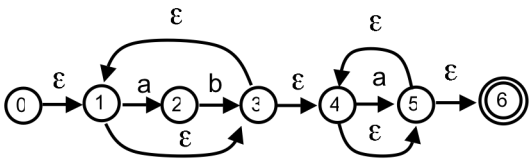
Homework 1 - Solution

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

a) 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)*.(a)*.
b) Do the sentences w_1 = "abaa" and w_2 = "aaa" belong to the language generated by this regular expression? Justify.
c) 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.
d) 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.

Answers:

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



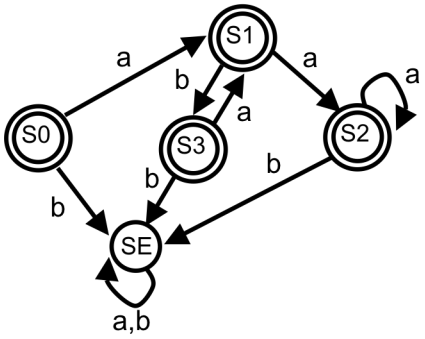
b) Both words are recognized by this NFA. Regarding the word "abaa" there is a path from state 0 to the accepting state 6, namely: 0,1,2,3,4,5,4,5,6. Regarding the word "aaa" the automaton may reach state 6 following the path 0,1,3,4,5,4,5,4,5,6.

c) Using the subset construction we arrive at the following subsets and transitions.

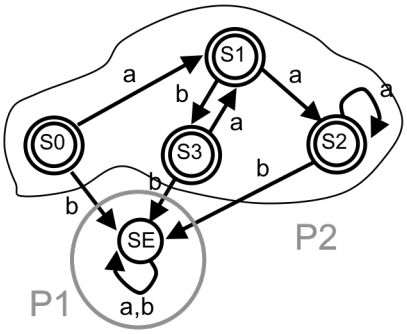
S0 = ε-closure (0) = {0, 1, 3, 4, 5, 6} – this is a final state because of state 6
S1 = DFAedge(S0,a) = ε-closure (goto(S0, a)) = {2, 4, 5, 6} – final state
SE = DFAedge(S0,b) = ε-closure (goto(S0, b)) = { }
S2 = DFAedge(S1,a) = ε-closure (goto(S1, a)) = {4, 5, 6} – final state
S3 = DFAedge(S1,b) = ε-closure (goto(S1, b)) = {3, 4, 5, 6} – final state

DFAedge(S2,a) = ε-closure (goto(S2, a)) = {4, 5, 6} = S2
DFAedge(S2,b) = ε-closure (goto(S2, b)) = { } = SE
DFAedge(S3,a) = ε-closure (goto(S3, a)) = {2, 4, 5, 6} = S1
DFAedge(S3,b) = ε-closure (goto(S3, b)) = { } = SE

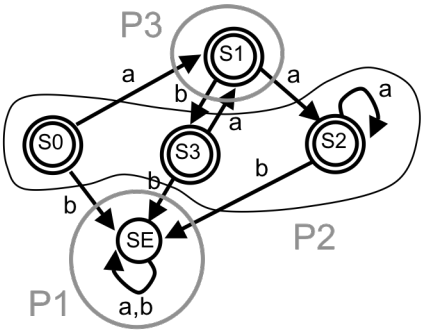
This results in the DFA shown below with starting state S0.



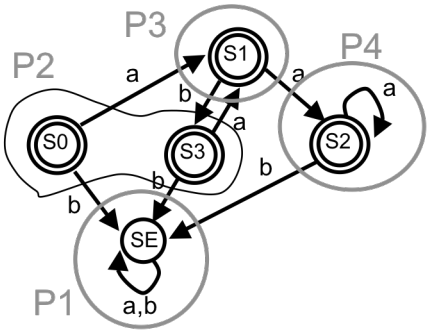
d) This DFA can be further minimize by using the iterative refinement partitioning yeilding the sequence of partitions indicated below.



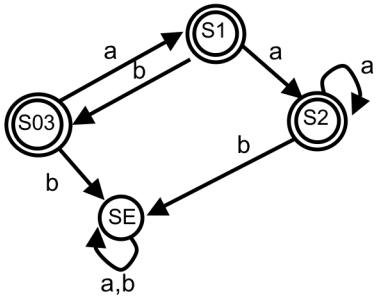
Initial partitioning on final states



Partitioning P2 on b

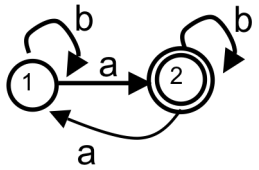


Partitioning P2 on a



final minimal DFA

Probem2 [30 points]: Consider the DFA below with starting state 1 and accepting state 2:



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

Answers:

a) 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 an sequence of "b"s of arbitrary length.
b) The derivations are shown below with the obvious symplification.

Expressions for k = 0
$R^0_{11} = (b|\varepsilon)$
$R^0_{12} = (a)$
$R^0_{21} = (a)$
$R^0_{22} = (b|\varepsilon)$

Expressions for k = 1
$R^1_{11} = R^0_{11} (R^0_{11})^* R^0_{11} \mid R^0_{11} = (b|\varepsilon) . (b|\varepsilon)^* . (b|\varepsilon) \mid (b|\varepsilon) = b^*$
$R^1_{12} = R^0_{11} (R^0_{11})^* R^0_{12} \mid R^0_{12} = (b|\varepsilon) . (b|\varepsilon)^* . (a) \mid (a) = (b|\varepsilon)^* (a) = b^*a$
$R^1_{21} = R^0_{21} (R^0_{11})^* R^0_{11} \mid R^0_{21} = (a) . (b|\varepsilon)^* . (b|\varepsilon) \mid (a) = ab^*$
$R^1_{22} = R^0_{21} (R^0_{11})^* R^0_{12} \mid R^0_{22} = (a) . (b|\varepsilon)^* . (a) \mid (b|\varepsilon) = a(b^*a \mid \varepsilon) \mid (a|\varepsilon) = ab^*a \mid b \mid \varepsilon$

Expressions for k = 2
$R^2_{11} = R^1_{12} (R^1_{22})^* R^1_{21} \mid R^1_{11} = (b^*a) . (ab^*a \mid b \mid \varepsilon)^* . ab^* \mid b^*$
$R^2_{12} = R^1_{12} (R^1_{22})^* R^1_{22} \mid R^1_{12} = (b^*a) . (ab^*a \mid b \mid \varepsilon)^* . (ab^*a \mid b \mid \varepsilon) \mid b^*a$
$R^2_{21} = R^1_{12} (R^1_{22})^* R^1_{21} \mid R^1_{21} = (b^*a) . (ab^*a \mid b \mid \varepsilon)^* . ab^* \mid ab^*$
$R^2_{22} = R^1_{12} (R^1_{22})^* R^1_{22} \mid R^1_{22} = (b^*a) . (ab^*a \mid b \mid \varepsilon)^* . (ab^*a \mid b \mid \varepsilon) \mid (ab^*a \mid b \mid \varepsilon)$

$L = R^2_{12} = (b^*a) . (ab^*a \mid b \mid \varepsilon)^* . (ab^*a \mid b \mid \varepsilon) \mid b^*a$

As can be seen the simplification of any of these regular expressions beyond the expresions for k=1 is fairly complicated. This method, although dorrect 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).

Answer: 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 ε-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 constructiona 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.

Answer: 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 reminder you would need no more that 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 S3. 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 S3 any additional 0 means you are shifting the bits by one bit, *i.e.,* multiplying by 2, hence staying in the same state.