Automata and Formal Languages — Homework 4

Due **Wednesday** 4th November 2015 (TA: Christopher Broadbent)

The lectures (slide 9 of *'implementing operations on sets'*) present a 'generic algorithm' $BinOp[\odot](A_1, A_2)$ paramaterised by a binary Boolean operation $\odot$. (For example, $\odot$ can be substituted for one of $\wedge$, $\vee$ or $\Leftrightarrow$ to produce a concrete algorithm for the substituting operation). The algorithm takes DFAs A_1 and A_2 as input and returns a DFA A such that $\mathcal{L}(A) = \mathcal{L}(A_1) \mathbin{\widehat{\odot}} \mathcal{L}(A_2)$ where

$$L_1 \mathbin{\widehat{\odot}} L_2 := \{ \ w \in \Sigma^* \mid (w \in L_1) \odot (w \in L_2) \ \}.$$

For example

$$L_1 \mathbin{\widehat{\wedge}} L_2 := \{ \ w \in \Sigma^* \mid (w \in L_1) \wedge (w \in L_2) \ \} = L_1 \cap L_2$$

Exercise 4.1

(a) Draw a three state DFA A_1 recognising ab^*.

(b) Draw a three state DFA A_2 recognising ba^*.

(c) Draw the DFA recognising $\mathcal{L}(A_1) \cup \mathcal{L}(A_2)$ that is computed by $BinOp[\vee](A_1, A_2)$.

Exercise 4.2

Describe concrete algorithms based on $BinOp[\odot](A_1, A_2)$ that achieve the following. You may need to modify the algorithm beyond merely substituting $\odot$, but the structure of your algorithms should be similar to $BinOp[\odot](A_1, A_2)$.

(a) An algorithm that takes two DFAs A_1 and A_2 as input and returns a DFA recognising $\overline{\mathcal{L}(A_1)} \cap \overline{\mathcal{L}(A_2)}$.

(b) An algorithm that takes a DFA A and a *reverse deterinistic* [1] NFA B that has at most one final state q_f, and returns a DFA recognising

$$\mathcal{L}(A) \cap \mathcal{L}(B)^R$$

where L^R denotes the *reverse* of L. [2]

(c) Let $\_ \, ? \, \_ \, : \, \_$ be the ternary Boolean operator whose truth table has entries of the following form:

b_1	b_2	b_3	$b_1 \, ? \, b_2 \, : \, b_3$
1	b_2	b_3	b_2
0	b_2	b_3	b_3

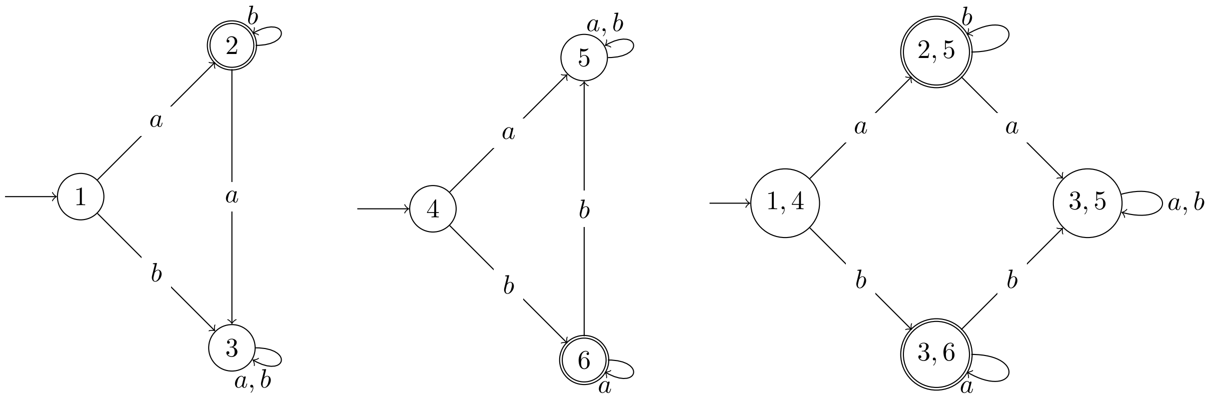
We would like an algorithm that takes three DFAs A_1, A_2 and A_3 as input and returns a DFA recognising

$$\{ \ w \in \Sigma^* \mid (w \in \mathcal{L}(A_1)) \, ? \, (w \in \mathcal{L}(A_2)) \, : \, (w \in \mathcal{L}(A_3)) \ \}$$

[1] An NFA $A = (Q, \Sigma, \delta, Q_0, F)$ is *reverse-deterministic* if $(q_1, a, q) \in \delta$ and $(q_2, a, q) \in \delta$ implies $q_1 = q_2$, i.e., no state has two input transitions labelled by the same letter.

[2] For a word $w = a_1 \cdots a_k$, the reverse of that word is defined by $w^R := a_k \cdots a_1$, and for a language $L \subseteq \Sigma^*$, the reverse of L is defined to be $L^R := \{ \ w^R \in \Sigma^* \mid w \in L \ \}$.

Solution 4.1



Solution 4.2

Input: DFAs $A_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ $A_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$
Output: DFA $A = (Q, \Sigma, \delta, Q_0, F)$ with $L(A) = \overline{L(A_1)} \cap \overline{L(A_2)}$

$Q, \delta, F \leftarrow \emptyset$
$q_0 \leftarrow [q_{01}, q_{02}]$
$W \leftarrow \{q_0\}$
while $W \neq \emptyset$ **do**
  pick $[q_1, q_2]$ **from** W
  add $[q_1, q_2]$ **to** Q
  if $(q_1 \notin F_1)$ and $(q_2 \notin F_2)$ **then add** $[q_1, q_2]$ **to** F
  for all $a \in \Sigma$ **do**
    $q_1' \leftarrow \delta_1(q_1, a)$; $q_2' \leftarrow \delta_2(q_2, a)$
    if $[q_1', q_2'] \notin Q$ **then add** $[q_1', q_2']$ **to** W
    add $([q_1, q_2], a, [q_1', q_2'])$ **to** δ

“

Input: DFA $A = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and reverse deterministic NFA $B = (Q_2, \Sigma, \delta_2, Q_{02}, \{ q_f \})$
Output: DFA $C = (Q, \Sigma, \delta, Q_0, F)$ with $L(C) = L(A) \cap L(B)^R$

$Q, \delta, F \leftarrow \emptyset$
$q_0 \leftarrow [q_{01}, q_f]$
$W \leftarrow \{q_0\}$
while $W \neq \emptyset$ **do**
  pick $[q_1, q_2]$ **from** W
  add $[q_1, q_2]$ **to** Q
  if $(q_1 \in F_1)$ and $(q_2 \in Q_{02})$ **then add** $[q_1, q_2]$ **to** F
  for all $a \in \Sigma$ **do**
    $q_1' \leftarrow \delta_1(q_1, a)$;
    $q_2' \leftarrow$ unique p s.t. $\delta_2(p, a) = q_2$
    if $[q_1', q_2'] \notin Q$ **then add** $[q_1', q_2']$ **to** W
    add $([q_1, q_2], a, [q_1', q_2'])$ **to** δ

Input: DFAs $A_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ $A_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ $A_3 = (Q_3, \Sigma, \delta_3, q_{03}, F_3)$
Output: DFA $A = (Q, \Sigma, \delta, Q_0, F)$ with $L(A) = L(A_1) \,\hat{?}\, L(A_2) \,\hat{:}\, L(A_3)$

$Q, \delta, F \leftarrow \emptyset$
$q_0 \leftarrow [q_{01}, q_{02}, q_{03}]$
$W \leftarrow \{q_0\}$
while $W \neq \emptyset$ **do**
  pick $[q_1, q_2, q_3]$ **from** W
  add $[q_1, q_2, q_3]$ **to** Q
  if $(q_1 \in F_1)\ ?\ (q_2 \in F_2)\ :\ (q_3 \in F_3)$ **then add** $[q_1, q_2]$ **to** F
  for all $a \in \Sigma$ **do**
    $q'_1 \leftarrow \delta_1(q_1, a)$; $q'_2 \leftarrow \delta_2(q_2, a)$; $q'_3 \leftarrow \delta_3(q_3, a)$;
    if $[q'_1, q'_2, q'_3] \notin Q$ **then add** $[q'_1, q'_2, q'_3]$ **to** W
    add $([q_1, q_2, q_3], a, [q'_1, q'_2, q'_3])$ **to** δ