Automata and Formal Languages — Homework 7

Due 02.12.2016

Exercise 7.1
Let $L_1 = \{abb, bba, bbb\}$ and $L_2 = \{aba, bbb\}$.

(a) Suppose you are given a fixed-length language L described explicitely by a set instead of an automaton. Give an algorithm that ouputs the state q of the master automaton for L.

(b) Use the previous algorithm to build the states of the master automaton for L_1 and L_2.

(c) Compute the state of the master automaton representing $L_1 \cup L_2$.

(d) Identify the kernels $\langle L_1 \rangle$, $\langle L_2 \rangle$, and $\langle L_1 \cup L_2 \rangle$.

Exercise 7.2

(a) Give an algorithm to compute $L(p) \cdot L(q)$ given states p and q of the master automaton.

(b) Give an algorithm to compute both the length and size of $L(q)$ given a state q of the master automaton.

(c) The length and size of $L(q)$ could be obtained in constant time if they were simply stored in the master automaton table. Give a new implementation of `make` for this representation.

Exercise 7.3
Let $k \in \mathbb{N}_{>0}$. Let flip : $\{0,1\}^k \to \{0,1\}^k$ be the function that inverts the bits of its input, e.g. flip(010) = 101. Let val : $\{0,1\}^k \to \mathbb{N}$ be such that val(w) is the number represented by w with the "least significant bit first" encoding.

(a) Describe the minimal transducer that accepts

$$L_k = \left\{[x, y] \in (\{0,1\} \times \{0,1\})^k : \mathrm{val}(y) = \mathrm{val}(\mathrm{flip}(x)) + 1 \bmod 2^k\right\} .$$

(b) Build the state r of the master transducer for L_3, and the state q of the master automaton for $\{010, 110\}$.

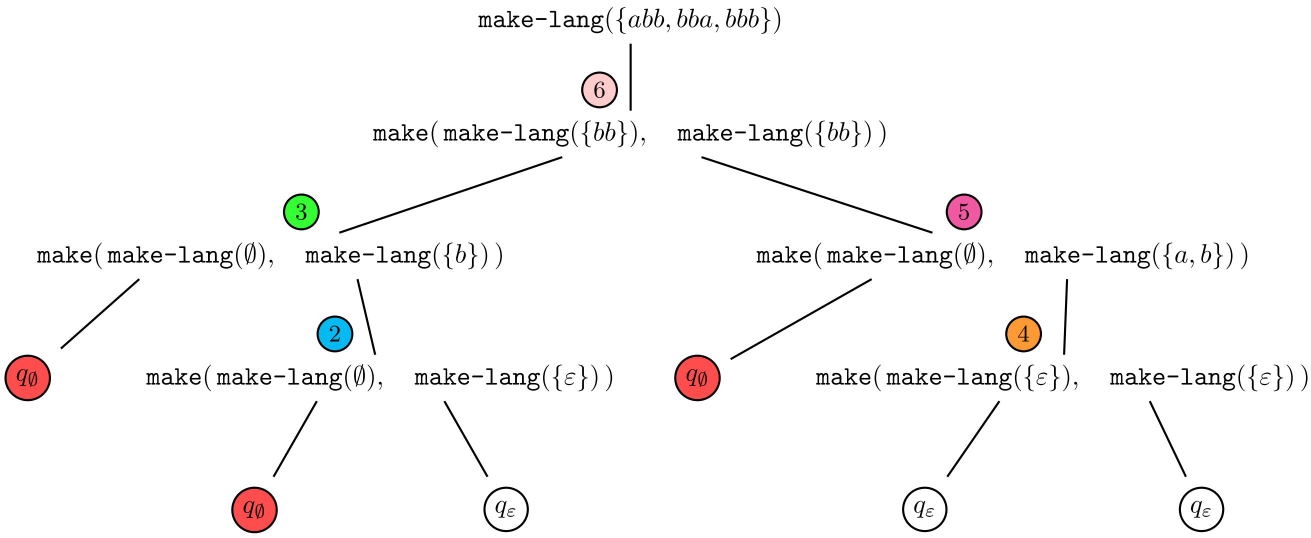
(c) Adapt the algorithm *pre* seen in class to compute *post*(r, q).

Solution 7.1

(a)

```
Input: Set of words L of fixed-length.
Output: state q of the master automaton such that L(q) = L.
make-lang(L):
    if L = ∅ then
        return q_∅
    else if L = {ε} then
        return q_ε
    else
        for a ∈ Σ do
            L^a ← {u : au ∈ L}
            s_a ← make-lang(L^a)
        return make(s)
```

(b) Executing make-lang(L_1) yields the following computation tree:



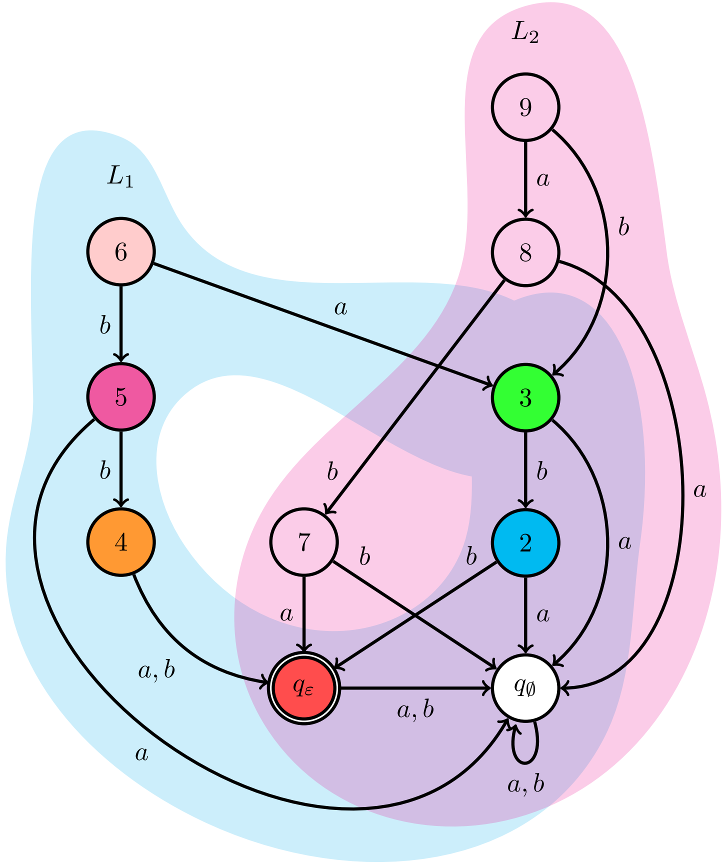
The table obtained after the execution is as follows:

Ident.	a-succ	b-succ
2	$q_\emptyset$	q_ε
3	$q_\emptyset$	2
4	q_ε	q_ε
5	$q_\emptyset$	4
6	3	5

Calling make-lang(L_2) adds the following rows to the table and returns 9:

Ident.	a-succ	b-succ
7	q_ε	$q_\emptyset$
8	$q_\emptyset$	7
9	8	3

The new master automaton fragment is:

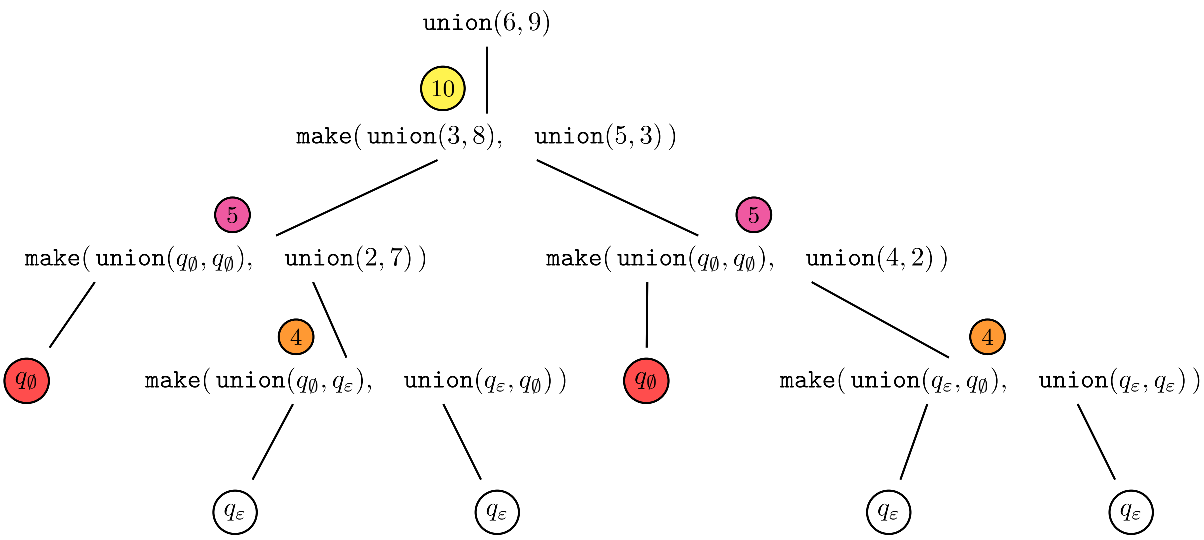


(c) We first adapt the algorithm for intersection to obtain an algorithm for union:

Input: states p, q of the master automaton with same length.
Output: state r of the master automaton such that $L(r) = L(p) \cup L(q)$.

```
union(p, q):
    if G(p, q) is not empty then
        return G(p, q)
    else if p = q_∅ and q = q_∅ then
        return q_∅
    else if p = q_ε or q = q_ε then
        return q_ε
    else
        for a ∈ Σ do
            s_a ← union(p^a, q^a)
        G(p, q) ← make(s)
        return G(p, q)
```

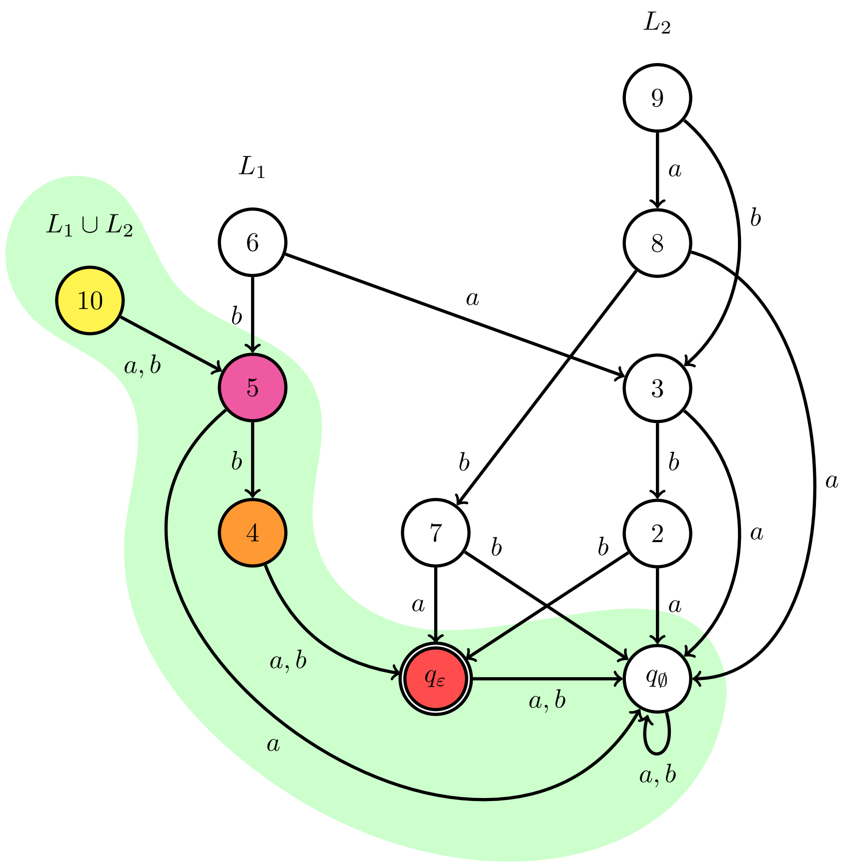
Executing `union`$(6, 9)$ yields the following computation tree:



Calling `union`$(6, 9)$ adds the following row to the table and returns 10:

Ident.	a-succ	b-succ
10	5	5

The new fragment of the master automaton is:



★ Note that `union` could be slightly improved by returning q whenever $p = q$, and updating $G(q, p)$ at the same time as $G(p, q)$.

(d) The kernels are:

$$\langle L_1 \rangle = L_1,$$
$$\langle L_2 \rangle = L_2,$$
$$\langle L_1 \cup L_2 \rangle = \{ba, bb\}.$$

Solution 7.2

(a) Let L, L' be fixed-length languages. We have

$$L \cdot L' = \begin{cases} \emptyset & \text{if } L = \emptyset, \\ L' & \text{if } L = \{\varepsilon\}, \\ \displaystyle\bigcup_{a \in \Sigma} a \cdot L^a \cdot L' & \text{otherwise.} \end{cases}$$

These identities give rise to the following algorithm:

Input: states p, q of the master automaton.
Output: state r of the master automaton such that $L(r) = L(p) \cdot L(q)$.

`concat`(L):
  if $G(p, q)$ is not empty **then**
    return $G(p, q)$
  else if $p = q_\emptyset$ **then**
    return $q_\emptyset$
  else if $p = q_\varepsilon$ **then**
    return q
  else
    for $a \in \Sigma$ **do**
      $s_a \leftarrow$ `concat`(p^a, q)
    $G(p, q) \leftarrow$ `make`(s)
    $G(q, p) \leftarrow G(p, q)$
    return $G(p, q)$

(b) Let L be a fixed-length language. We have

$$\text{length}(L) = \begin{cases} \infty & \text{if } L = \emptyset, \\ 0 & \text{if } L = \{\varepsilon\}, \\ \text{length}(L^a) + 1 \text{ for any } a \in \Sigma \text{ s.t. } L^a \neq \emptyset & \text{otherwise.} \end{cases}$$

and

$$|L| = \begin{cases} 0 & \text{if } L = \emptyset, \\ 1 & \text{if } L = \{\varepsilon\}, \\ \sum_{a \in \Sigma} |L^a| & \text{otherwise.} \end{cases}$$

These identities give rise to the following algorithm:

Input: state p of the master automaton.
Output: length and size of $L(q)$.

```
len-size(q):
    if G(q) is not empty then
        return G(q)
    else if q = q_∅ then
        return (∞, 0)
    else if q = q_ε then
        return (0, 1)
    else
        k ← ∞
        n ← 0
        for a ∈ Σ do
            k′, n′ ← len-size(q^a)
            if k′ ≠ ∞ then k ← max(k, k′) + 1
            n ← n + n′
        G(q) ← (k, n)
        return G(q)
```

(c) Let q be a state of the master automaton. We denote the length and the size of q respectively by $\text{len}(q)$ and $|q|$. These values are encoded in two new columns of the master automaton table. We set

$$\begin{aligned} \text{len}(q_\emptyset) &= \infty, & |q_\emptyset| &= 0. \\ \text{len}(q_\varepsilon) &= 0, & |q_\varepsilon| &= 1. \end{aligned}$$

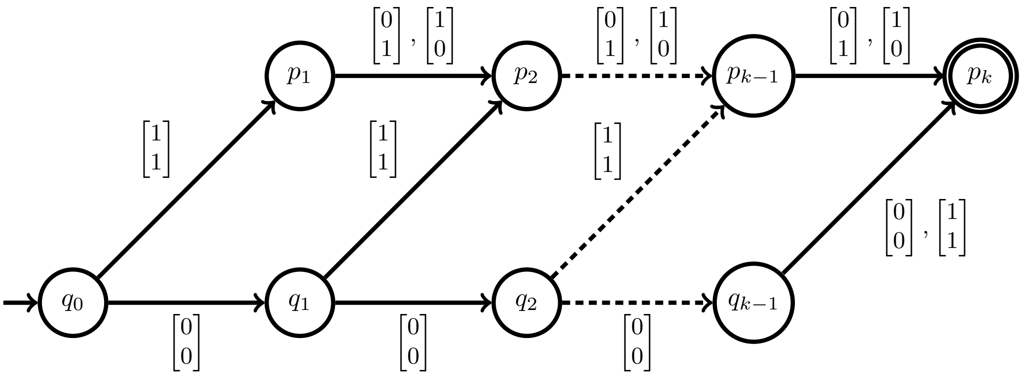
From the observations made in the previous question, we obtain the following algorithm:

Input: mapping s from Σ to the master automaton states.
Output: state q such that $L(q)^a = s_a$ for every $a \in \Sigma$.

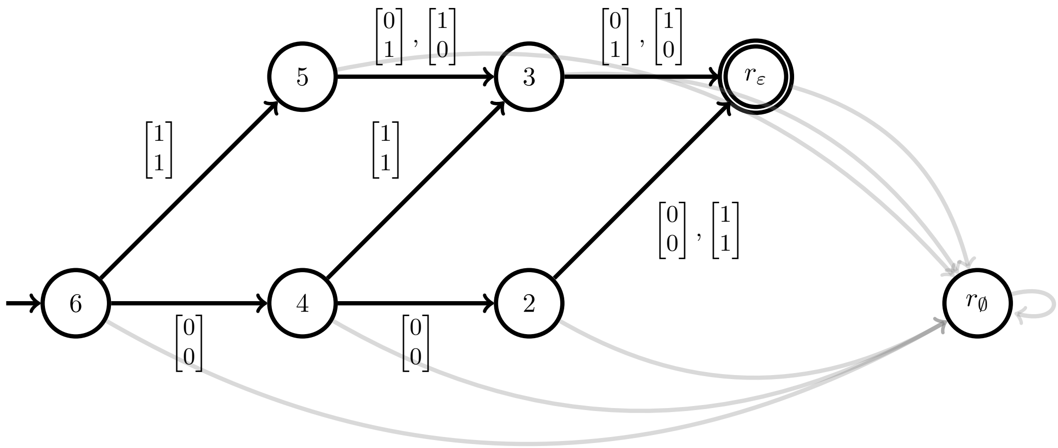
```
make'(q):
    q_max ← 0
    for row q, t ∈ Table do
        if s = t then
            return q
        else
            q_max ← max(q_max, q).
    r ← q_max + 1

    k ← ∞                                   /* Compute length and size */
    n ← 0
    for a ∈ Σ do
        if s_a ≠ q_∅ then k ← |s_a| + 1
        n ← n + len(s_a)

    Table(r) ← (s, k, n)
    return r
```

Solution 7.3

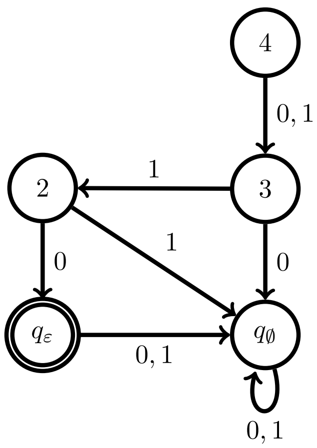
(a) Let $[x, y] \in L_k$. We may flip the bits of x at the same time as adding 1. If $x_1 = 1$, then $\neg x = 0$, and hence adding 1 to $\text{val}(\text{flip}(x))$ results in $y_1 = 1$. Thus, for every $1 < i \leq k$, we have $y_i = \neg x_i$. If $x_1 = 0$, then $\neg x_1 = 1$. Adding 1 yields $y_1 = 0$ with a carry. This carry is propagated as long as $\neg x_i = 1$, and thus as long as $x_i = 0$. When some position j with $x_j = 1$ is encountered, the carry is "consumed", and we flip the remaining bits of x. These observations give rise to the following minimal transducer for L_k:



(b) The minimal transducer accepting L_3 is



State 4 of the following fragment of the master automaton accepts $\{010, 110\}$:



(c) We can establish the following identities similar to those obtained for *pre*:

$$post_R(L) = \begin{cases} \emptyset & \text{if } R = \emptyset \text{ or } L = \emptyset, \\ \{\varepsilon\} & \text{if } R = \{[\varepsilon, \varepsilon]\} \text{ and } L = \{\varepsilon\}, \\ \displaystyle\bigcup_{a,b \in \Sigma} b \cdot post_{R^{[a,b]}}(L^a) & \text{otherwise.} \end{cases}$$

To see that these identities hold, let $b \in \Sigma$ and $v \in \Sigma^k$ for some $k \in \mathbb{N}$. We have,

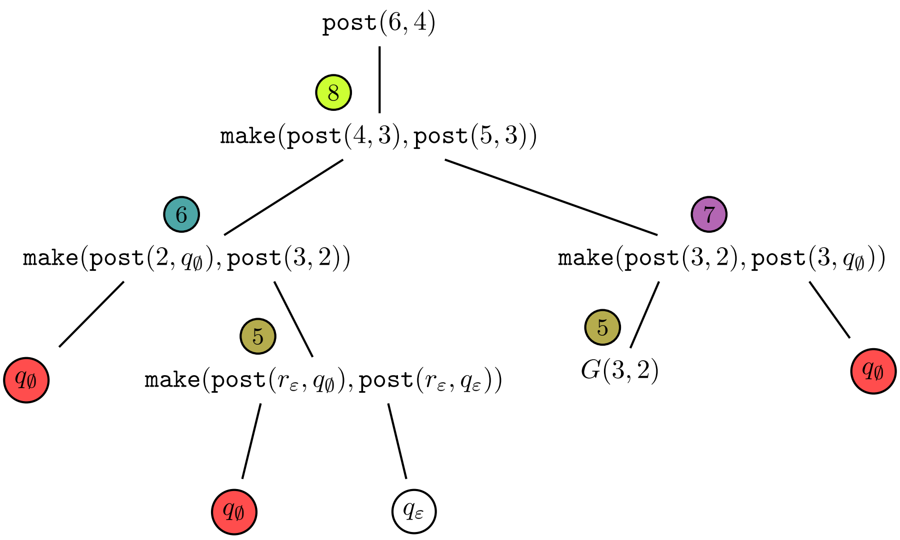
$$
\begin{aligned}
bv \in \mathit{post}_R(L) &\iff \exists a \in \Sigma, u \in \Sigma^k \text{ s.t. } au \in L \text{ and } [au, bv] \in R \\
&\iff \exists a \in \Sigma, u \in L^a \text{ s.t. } [au, bv] \in R \\
&\iff \exists a \in \Sigma, u \in L^a \text{ s.t. } [u, v] \in R^{[a,b]} \\
&\iff \exists a \in \Sigma \text{ s.t. } v \in \mathit{Post}_{R^{[a,b]}}(L^a) \\
&\iff v \in \bigcup_{a \in \Sigma} \mathit{Post}_{R^{[a,b]}}(L^a) \\
&\iff bv \in \bigcup_{a \in \Sigma} b \cdot \mathit{Post}_{R^{[a,b]}}(L^a).
\end{aligned}
$$

We obtain the following algorithm:

Input: state r of the master transducer and state q of the master automaton.
Output: $\mathit{Post}_R(L)$ where $R = L(r)$ and $L = L(q)$.

```
1  post(r, q):
2      if G(r, q) is not empty then
3          return G(r, q)
4      else if r = r_∅ or q = q_∅ then
5          return q_∅
6      else if r = r_ε and q = q_ε then
7          return q_ε
8      else
9          for b ∈ Σ do
10             p ← q_∅
11             for a ∈ Σ do
12                 p ← union(p, post(r^[a,b], q^a))
13             s_b ← p
14         G(q, r) ← make(s)
15         return G(q, r)
```

Note that the transducer for L_3 has some "strong" deterministic property. Indeed, for every state r and $b \in \{0, 1\}$, if $r^{[a,b]} \neq r_\emptyset$ then $r^{[\neg a,b]} = r_\emptyset$. Hence, for a fixed $b \in \{0, 1\}$, at most one $\texttt{post}(r^{[a,b]}, q^a)$ can differ from $q_\emptyset$ at line 12 of the algorithm. Thus, unions made by the algorithm on this transducer are trivial, and executing $\texttt{post}(6, 4)$ yields the following computation tree:



Calling `post`$(6, 4)$ adds the following rows to the master automaton table and returns 8:

Ident.	0-succ	1-succ
5	$q_\emptyset$	q_ε
6	$q_\emptyset$	5
7	5	$q_\emptyset$
8	6	7

The new master automaton fragment: