

Automata and Formal Languages — Homework 8

Due 11.12.2018

Exercise 8.1

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

(a) Give an algorithm for the following operation:

INPUT: A fixed-length language $L \subseteq \Sigma^k$ described explicitly as a set of words.
OUTPUT: State q of the master automaton over Σ such that $L(q) = L$.

(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 8.2

(a) Give an recursive algorithm for the following operation:

INPUT: States p and q of the master automaton.
OUTPUT: State r of the master automaton such that $L(r) = L(p) \cdot L(q)$.

Observe that the languages $L(p)$ and $L(q)$ can have different lengths. Try to reduce the problem for p, q to the problem for p^a, q .

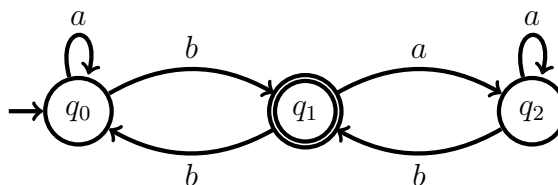
(b) Give an recursive algorithm for the following operation:

INPUT: A state q of the master automaton.
OUTPUT: State r of the master automaton such that $L(r) = L(q)^R$.

(c) Give an recursive algorithm for the following operation:

INPUT: A DFA A over alphabet Σ , and $k \in \mathbb{N}$.
OUTPUT: State q of the master automaton over Σ such that $L(q) = L(A) \cap \Sigma^k$.

Apply your algorithm on the following DFA with $k = 3$:



Exercise 8.3

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

- (a) Describe the minimal transducer that accepts

$$L_k = \{[x, y] \in (\{0, 1\} \times \{0, 1\})^k \mid \text{val}(y) = \text{val}(\text{flip}(x)) + 1 \pmod{2^k}\}.$$

- (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* and compute using this algorithm $\text{post}(r, q)$.

Solution 8.1

(a)

Input: A fixed-length language $L \subseteq \Sigma^k$ described explicitly by a set of words.

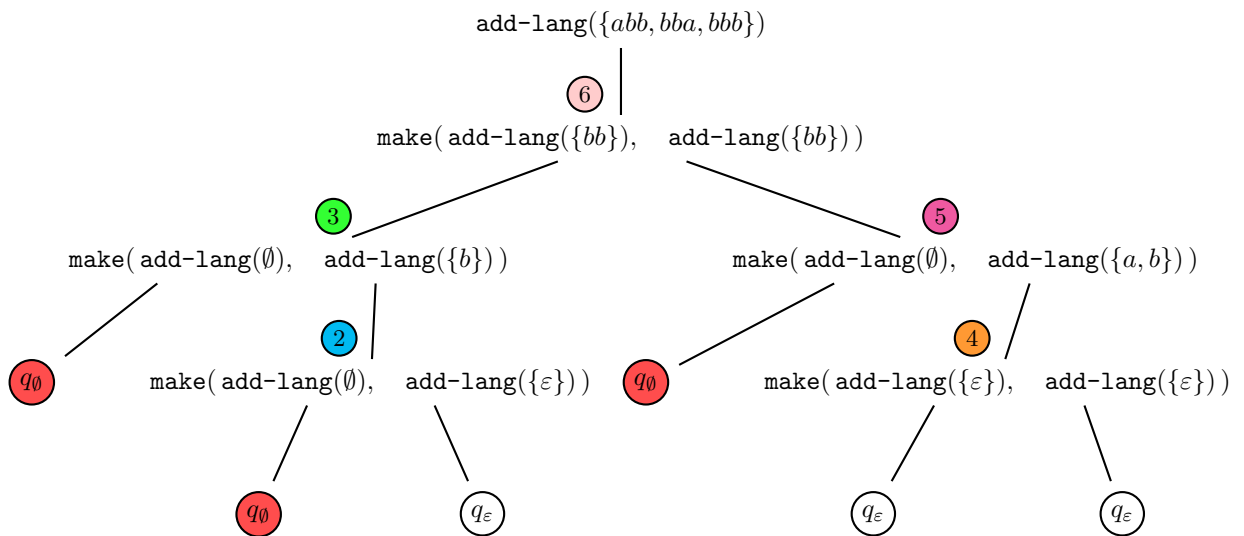
Output: State q of the master automaton over Σ such that $L(q) = L$.

```

1 add-lang(L) :
2   if L = ∅ then
3     return q∅
4   else if L = {ε} then
5     return qε
6   else
7     for ai ∈ Σ do
8       Lai ← {u | aiu ∈ L}
9       si ← add-lang(Lai)
10    return make(s1, s2, ..., sn)

```

(b) Executing `add-lang(L1)` yields the following computation tree:



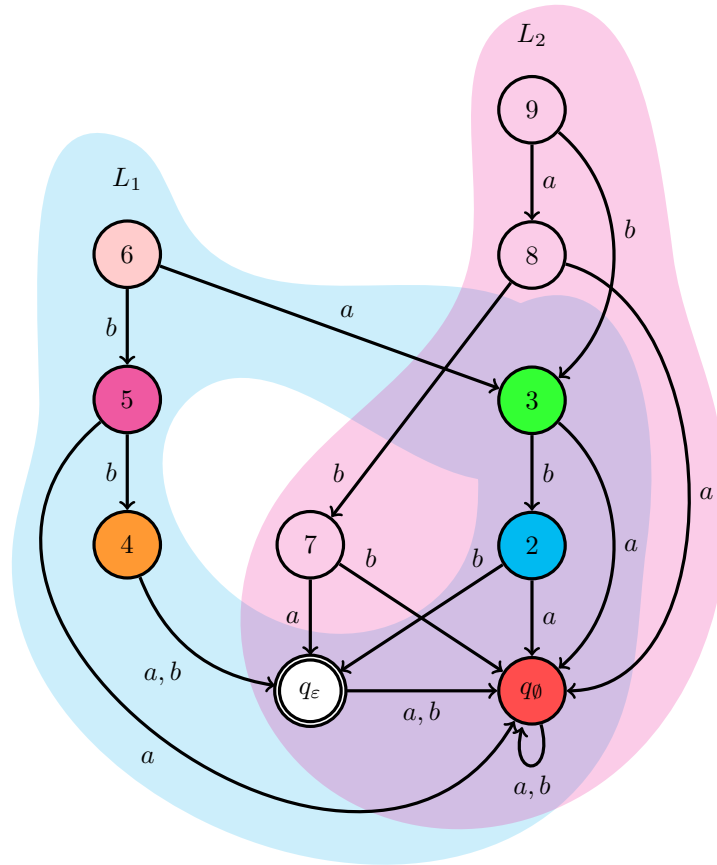
The table obtained after the execution is as follows:

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 `add-lang(L2)` 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 resulting master automaton fragment is:



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

Input: States p and q of same length of the master automaton.

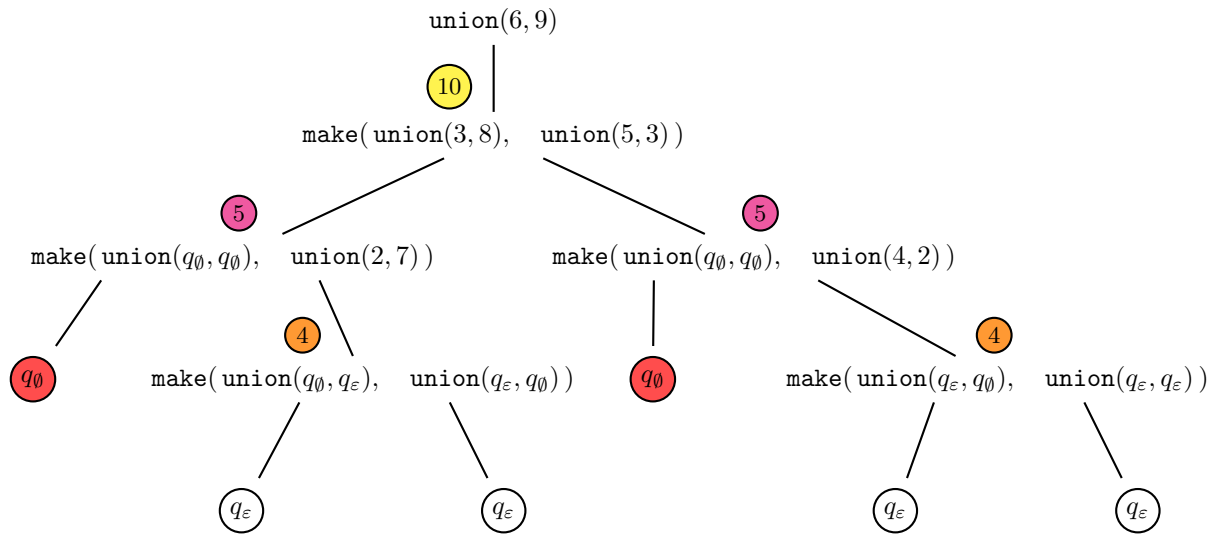
Output: State r of the master automaton such that $L(r) = L(p) \cup L(q)$.

```

1 union( $p, q$ ) :
2   if  $G(p, q)$  is not empty then
3     return  $G(p, q)$ 
4   else if  $p = q_\emptyset$  and  $q = q_\emptyset$  then
5     return  $q_\emptyset$ 
6   else if  $p = q_\varepsilon$  or  $q = q_\varepsilon$  then
7     return  $q_\varepsilon$ 
8   else
9     for  $a_i \in \Sigma$  do
10       $s_i \leftarrow \text{union}(p^{a_i}, q^{a_i})$ 
11       $G(p, q) \leftarrow \text{make}(s_1, s_2, \dots, s_n)$ 
12      return  $G(p, q)$ 

```

Executing $\text{union}(6, 9)$ yields the following computation tree:



Calling $\text{union}(6, 9)$ 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 by updating $G(q, p)$ at the same time as $G(p, q)$.

(d) The kernels are:

$$\begin{aligned}\langle L_1 \rangle &= L_1, \\ \langle L_2 \rangle &= L_2, \\ \langle L_1 \cup L_2 \rangle &= \{ba, bb\}.\end{aligned}$$

Solution 8.2

(a) Let L and L' be fixed-length languages. The following holds:

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

These identities give rise to the following algorithm:

Input: States p and q of the master automaton.
Output: State r of the master automaton such that $L(r) = L(p) \cdot L(q)$.

```

1 concat( $p, q$ ):
2   if  $G(p, q)$  is not empty then
3     return  $G(p, q)$ 
4   else if  $p = q_\emptyset$  then
5     return  $q_\emptyset$ 
6   else if  $p = q_\varepsilon$  then
7     return  $q$ 
8   else
9     for  $a_i \in \Sigma$  do
10       $s_i \leftarrow \text{concat}(p^{a_i}, q)$ 
11       $G(p, q) \leftarrow \text{make}(s_1, s_2, \dots, s_n)$ 
12     return  $G(p, q)$ 

```

(b) Let L be a fixed-length language. The following holds:

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

These identities give rise to the following algorithm:

(c) Let A be a DFA and let $k \in \mathbb{N}$. The following holds:

$$L(A) \cap \Sigma^k = \begin{cases} \emptyset & \text{if } k = 0 \text{ and } \varepsilon \notin L(A), \\ \{\varepsilon\} & \text{if } k = 0 \text{ and } \varepsilon \in L(A), \\ \bigcup_{a \in \Sigma} \{a\} \cdot (L(A)^a \cap \Sigma^{k-1}) & \text{otherwise.} \end{cases}$$

These identities give rise to the following algorithm:

Input: A state q of the master automaton.

Output: State r of the master automaton such that $L(r) = L(q)^R$.

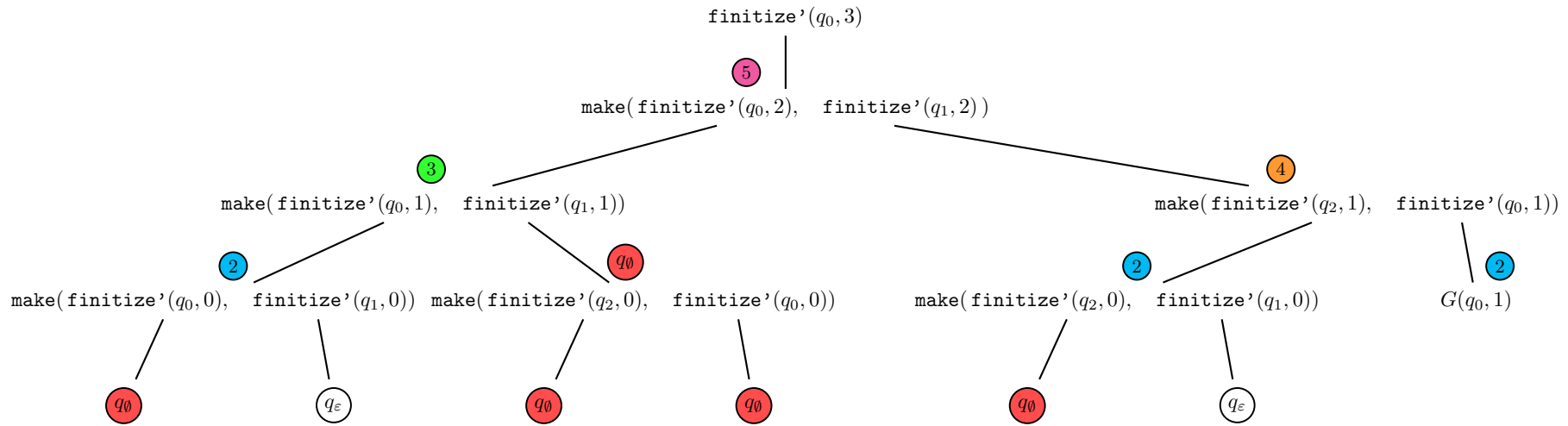
```
1 reverse( $q$ ):
2   if  $G(q)$  is not empty then
3     return  $G(q)$ 
4   else if  $q = q_\emptyset$  then
5     return  $q_\emptyset$ 
6   else if  $q = q_\varepsilon$  then
7     return  $q_\varepsilon$ 
8   else
9      $p \leftarrow q_\emptyset$ 
10    for  $a_i \in \Sigma$  do
11       $s_i \leftarrow q_\varepsilon$ 
12       $s_j \leftarrow q_\emptyset$  for every  $i \neq j$ 
13       $r \leftarrow \text{concat}(\text{reverse}(q^{a_i}), \text{make}(s_1, s_2, \dots, s_n))$ 
14       $p \leftarrow \text{union}(p, r)$ 
15     $G(q) \leftarrow p$ 
16    return  $G(q)$ 
```

Input: A DFA A over alphabet Σ , and $k \in \mathbb{N}$.

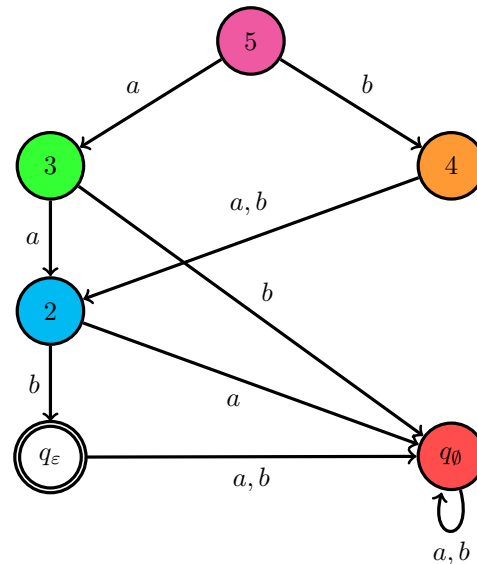
Output: State q of the master automaton over Σ such that $L(q) = L(A) \cap \Sigma^k$.

```
1 finitize( $A, k$ ):
2    $(Q, q_0, \Sigma, \delta, F) \leftarrow A$ 
3   return finitize'( $q_0, k$ )
4
5 finitize'( $q, k$ ):
6   if  $G(q, k)$  is not empty then
7     return  $G(q, k)$ 
8   else if  $k = 0$  and  $q \notin F$  then
9     return  $q_\emptyset$ 
10  else if  $k = 0$  and  $q \in F$  then
11    return  $q_\varepsilon$ 
12  else
13    for  $a_i \in \Sigma$  do
14       $s_i \leftarrow \text{finitize}'(\delta(q, a_i), k - 1)$ 
15       $G(q, k) \leftarrow \text{make}(s_1, s_2, \dots, s_n)$ 
16    return  $G(q, k)$ 
```

Executing $\text{finitize}(A, 3)$ calls $\text{finitize}'(q_0, 3)$ which yields the following computation tree:

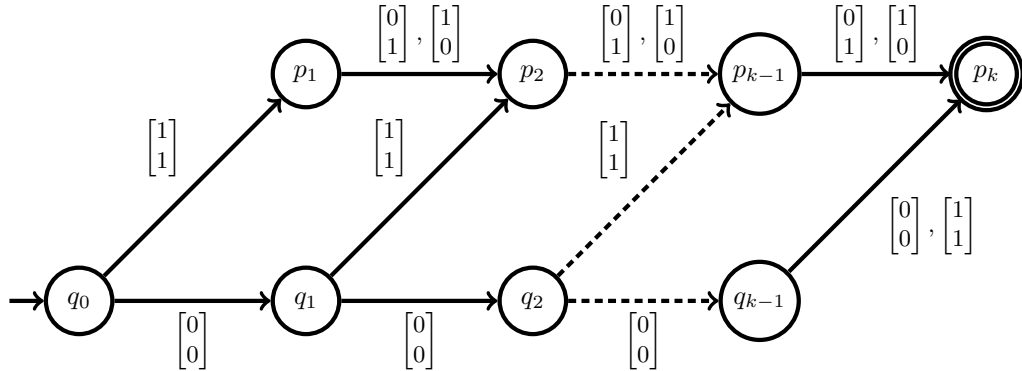


State 5 of the following master automaton fragment accepts $L(A) \cap \{a, b\}^3 = \{aab, bab, bbb\}$:

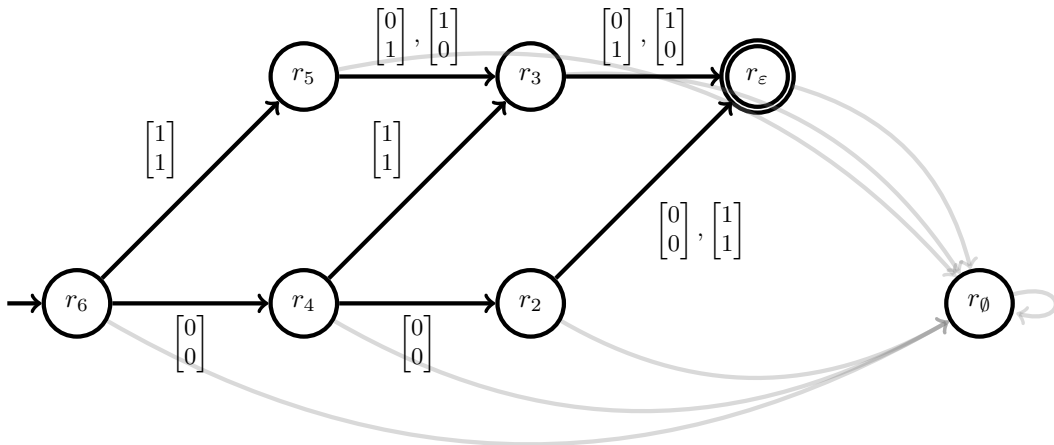


Solution 8.3

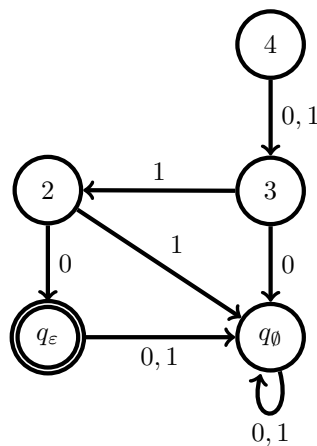
- (a) Let $[x, y] \in L_k$. We may flip the bits of x at the same time as adding 1. If $x_1 = 1$, then $\neg x_1 = 0$, and hence adding 1 to $\text{val}(\text{flip}(x))$ results in $y_1 = 1$. Thus, for every $1 < i \leq k$, we have $y_i = \neg x_i$. If $x_1 = 0$, then $\neg x_1 = 1$. Adding 1 yields $y_1 = 0$ with a carry. This carry is propagated as long as $\neg x_i = 1$, and thus as long as $x_i = 0$. If 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 master automaton fragment accepts $\{010, 110\}$:



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

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

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

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

We obtain the following algorithm:

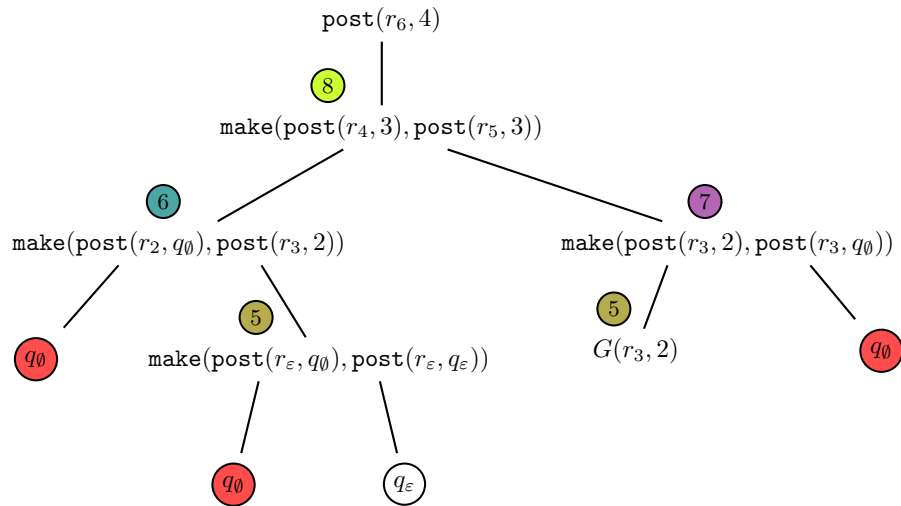
Input: A state r of the master transducer and a state q of the master automaton.
Output: State p of the master automaton such that $L(p) = Post_R(L)$ where $R = L(r)$ and $L = L(q)$.

```

1 post( $r, q$ ):
2   if  $G(r, q)$  is not empty then
3     return  $G(r, q)$ 
4   else if  $r = r_\emptyset$  or  $q = q_\emptyset$  then
5     return  $q_\emptyset$ 
6   else if  $r = r_\varepsilon$  and  $q = q_\varepsilon$  then
7     return  $q_\varepsilon$ 
8   else
9     for  $b_i \in \Sigma$  do
10       $p \leftarrow q_\emptyset$ 
11      for  $a \in \Sigma$  do
12         $p \leftarrow \text{union}(p, \text{post}(r^{[a,b_i]}, q^a))$ 
13       $s_i \leftarrow p$ 
14       $G(q, r) \leftarrow \text{make}(s_1, s_2, \dots, s_n)$ 
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 $r^{[-a,b]} = r_\emptyset$. Hence, for a fixed $b \in \{0, 1\}$, at most one term of the form “ $\text{post}(r^{[a,b]}, q^a)$ ” can differ from q_\emptyset at line 12 of the algorithm. Thus, unions made by the algorithm on this transducer are trivial, and executing $\text{post}(6, 4)$ yields the following computation tree:



Calling $\text{post}(6, 4)$ 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 resulting master automaton fragment:

