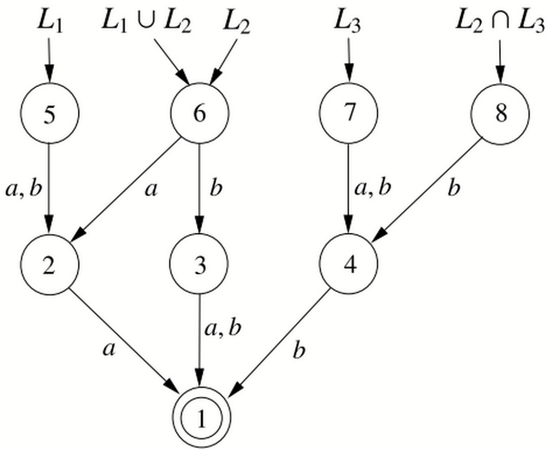
Implementing union and intersection



- We give a recursive algorithm $inter[T](q_1, q_2)$:
  - Input: state identifiers q_1, q_2 from table T.
  - Output: identifier of the state recognizing $L(q_1) \cap L(q_2)$ in the multi-DFA for T.
  - Side-effect: if the identifier is not in T, then the algorithm adds new nodes to T, i.e., after termination the table T may have been extended.
- The algorithm follows immediately from the following properties

  (1) if $L_1 = \emptyset$, then $L_1 \cap L_2 = \emptyset$;

  (2) if $L_2 = \emptyset$, then $L_1 \cap L_2 = \emptyset$;

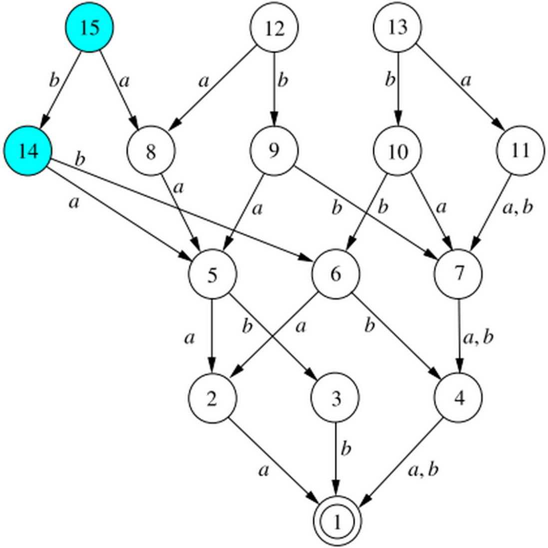
  (3) If $L_1 \neq \emptyset$ and $L_2 \neq \emptyset$, then $(L_1 \cap L_2)^a = L_1^a \cap L_2^a$ for every $a \in \Sigma$.

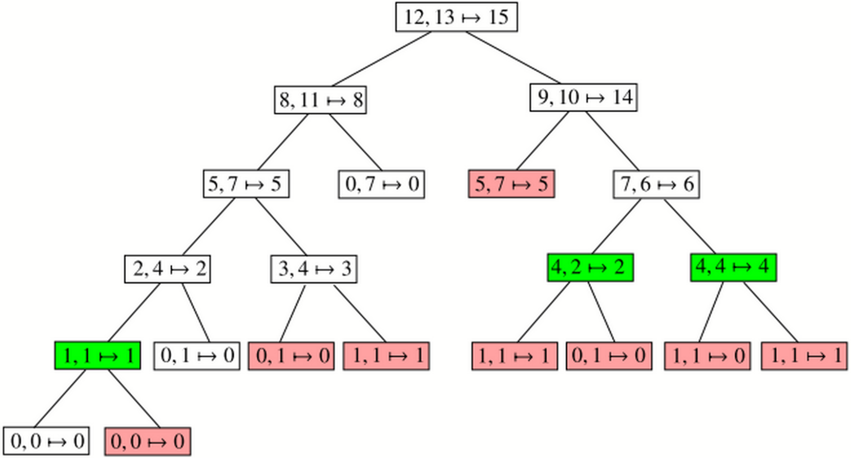
$inter[T](q_1, q_2)$

Input: table T, states q_1, q_2 of T

Output: state recognizing $\mathcal{L}(q_1) \cap \mathcal{L}(q_2)$

if $G(q_1, q_2)$ is not empty **then return** $G(q_1, q_2)$
if $q_1 = q_\emptyset \vee q_2 = q_\emptyset$ **then return** $q_\emptyset$
if $q_1 \neq q_\emptyset \wedge q_2 \neq q_\emptyset$ **then**
  for all $i = 1, \ldots, m$ **do** $r_i \leftarrow inter[T](q_1^{a_i}, q_2^{a_i})$
  $G(q_1, q_2) \leftarrow \texttt{make}[T](r_1, \ldots, r_m)$
  return $G(q_1, q_2)$


15
12
13
b
a
a
b
b
a
14
8
9
10
11
b
a
a
a
b
b
a
a, b
5
6
7
a
b
a
b
a, b
2
3
4
a
b
a, b
1


12, 13 ↦ 15
8, 11 ↦ 8
9, 10 ↦ 14
5, 7 ↦ 5
0, 7 ↦ 0
5, 7 ↦ 5
7, 6 ↦ 6
2, 4 ↦ 2
3, 4 ↦ 3
4, 2 ↦ 2
4, 4 ↦ 4
1, 1 ↦ 1
0, 1 ↦ 0
0, 1 ↦ 0
1, 1 ↦ 1
1, 1 ↦ 1
0, 1 ↦ 0
1, 1 ↦ 0
1, 1 ↦ 1
0, 0 ↦ 0
0, 0 ↦ 0

Fixed-length complement

In principle ill-defined, because the complement of a fixed-length language is not fixed-length.

We implement the fixed-length complement instead.

Can't we just swap the states for the empty language and the language containing the empty word?

Yes and no ...

Fixed-length complement

Equations:

- if $L = \emptyset$, then $\overline{L} = \Sigma^n$, where n is the length of L;
- if $L = \{\epsilon\}$, then $\overline{L} = \emptyset$; and
- if $\emptyset \neq L \neq \{\epsilon\}$, then $\left(\overline{L}\right)^a = \overline{L^a}$.

  (Observe that $w \in \left(\overline{L}\right)^a$ iff $aw \notin L$ iff $w \notin L^a$ iff $w \in \overline{L^a}$.)

$comp[T,n](q)$

Input: table T, length n, state q of T of length n

Output: state recognizing the fixed-length complement of $L(q)$

if $G(q)$ is not empty **then return** $G(q)$

if $n = 0$ **and** $q = q_\emptyset$ **then return** q_ϵ

else if $n = 0$ **and** $q = q_\epsilon$ **then return** $q_\emptyset$

else $/ * \; n \geq 1 \; * /$

 for all $i = 1, \ldots, m$ **do** $r_i \leftarrow comp[T, n-1](q^{a_i})$

 $G(q) \leftarrow \mathtt{make}[T](r_1, \ldots, r_m)$

 return $G(q)$

Emptiness

empty[T](q)
Input: table T, state q of T
Output: **true** if $\mathcal{L}(q) = \emptyset$, **false** otherwise

```
return q = q_∅
```

Universality

- if $L = \emptyset$, then L is not universal;
- if $L = \{\epsilon\}$, then L is universal;
- if $\emptyset \neq L \neq \{\epsilon\}$, then L is universal iff L^a is universal for every $a \in \Sigma$.

$univ[T](q)$

Input: table T, state q of T

Output: **true** if $\mathcal{L}(q)$ is fixed-length universal,
false otherwise

if $G(q)$ is not empty **then return** $G(q)$
if $q = q_\emptyset$ **then return false**
else if $q = q_\epsilon$ **then return true**
else $/* \; q \neq q_\emptyset$ and $q \neq q_\epsilon \; */$
  for all $i = 1, \ldots, m$ **do** $r_i \leftarrow comp[T](q^{a_i})$
  $G(q) \leftarrow$ **and**$(univ[T](r_1), \ldots, univ[T](r_m))$
  return $G(q)$

Inclusion and Equality

Inclusion. Given two languages $L_1, L_2 \subseteq \Sigma^n$, in order to check $L_1 \subseteq L_2$ we compute $L_1 \cap L_2$ and check whether it is equal to L_1 using the equality check shown next. The complexity is dominated by the complexity of computing the intersection.

$eq[T](q_1, q_2)$
Input: table T, states q_1, q_2 of T
Output: **true** if $\mathcal{L}(q_1) = \mathcal{L}(q_2)$, **false** otherwise

```
1   return q1 = q2
```

$eq[T_1, T_2](q_1, q_2)$

Input: tables T_1, T_2, states q_1 of T_1, q_2 of T_2

Output: **true** if $\mathcal{L}(q_1) = \mathcal{L}(q_2)$, **false** otherwise

if $G(q_1, q_2)$ is not empty **then return** $G(q_1, q_2)$
if $q_1 = q_{\emptyset 1}$ and $q_2 = q_{\emptyset 2}$ **then** $G(q_1, q_2) \leftarrow \texttt{true}$
else if $q_1 = q_{\emptyset 1}$ and $q_2 \neq q_{\emptyset 2}$ **then** $G(q_1, q_2) \leftarrow \texttt{false}$
else if $q_1 \neq q_{\emptyset 1}$ and $q_2 = q_{\emptyset 2}$ **then** $G(q_1, q_2) \leftarrow \texttt{false}$
else $/* \; q_1 \neq q_{\emptyset 1}$ and $q_2 \neq q_{\emptyset 2} \; */$
    $G(q_1, q_2) \leftarrow$ **and**$(\mathsf{eq}(q_1^{a_1}, q_2^{a_1}), \ldots, \mathsf{eq}(q_1^{a_m}, q_2^{a_m}))$
return $G(q_1, q_2)$

What if the starting point is an NFA?

- Given: NFA A accepting a fixed-length language and containing no cycles.

  Goal: simultaneously determinize and minimize A
- Each state of A accepts a fixed-length language.
- We give an algorithm *state*(S):
  - Input: a subset S of states of A accepting languages of the same length.
  - Output: the state of the master automaton accepting $\bigcup_{q \in S} L(q)$.
- Goal is achieved by calling *state*($\{q_0\}$)

Equations:

- if $S = \emptyset$ then $\mathcal{L}(S) = \emptyset$;
- if $S \cap F \neq \emptyset$ then $\mathcal{L}(S) = \{\epsilon\}$
- if $S \neq \emptyset$ and $S \cap F = \emptyset$, then $\mathcal{L}(S) = \bigcup_{i=1}^{n} a_i \cdot \mathcal{L}(S_i)$, where $S_i = \delta(S, a_i)$.

$\mathit{state}[A](S)$

Input: NFA $A = (Q, \Sigma, \delta, q_0, F)$, set $S \subseteq Q$

Output: master state recognizing $\mathcal{L}(S)$

if $G(S)$ is not empty **then return** $G(S)$

else if $S = \emptyset$ **then return** $q_\emptyset$

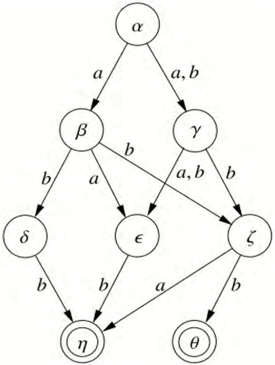
else if $S \cap F \neq \emptyset$ **then return** q_ϵ

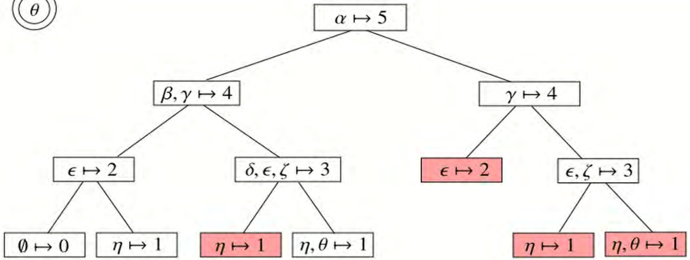
else $/* S \neq \emptyset$ and $S \cap F = \emptyset */$

  for all $i = 1, \ldots, m$ **do** $S_i \leftarrow \delta(S, a_i)$

  $G(S) \leftarrow \mathit{make}(\mathit{state}[A](S_1), \ldots, \mathit{state}[A](S_m))$;

  return $G(S)$


α
a
a, b
β
γ
b
b
a
a, b
b
δ
ϵ
ζ
b
b
a
b
η
θ


α ↦ 5
β, γ ↦ 4
γ ↦ 4
ϵ ↦ 2
δ, ϵ, ζ ↦ 3
ϵ ↦ 2
ϵ, ζ ↦ 3
∅ ↦ 0
η ↦ 1
η ↦ 1
η, θ ↦ 1
η ↦ 1
η, θ ↦ 1