## Remarks:

1. Complement and then check for emptiness
   - exponential complexity
2. Possible improvements:
   - check for emptiness while complementing: on-the-fly-check
   - test for subsumption

# A Subsumption Test

We observe that, while doing the conversion to and the universality check for a DFA, it might not be necessary to store all states.

**Definition 32**

Let $A$ be a NFA, and let $B = \text{NFAtoDFA}(A)$. A state $Q'$ of $B$ is minimal if no other state $Q''$ of $B$ satisfies $Q'' \subset Q'$.

**Lemma 33**

*Let $A$ be an NFA, and let $B = \text{NFAtoDFA}(A)$. $A$ is universal iff every minimal state of $B$ is final.*

Proof.

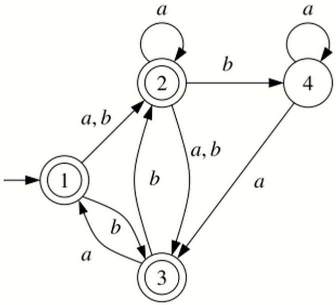
Since $A$ and $B$ recognize the same language, $A$ is universal iff $B$ is universal. So $A$ is universal iff every state of $B$ is final. But a state of $B$ is final iff it contains some final state of $A$, and so every state of $B$ is final iff every minimal state of $B$ is final. $\square$

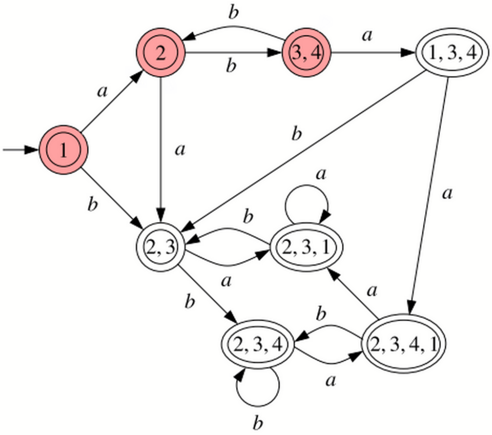
*UnivNFA*$(A)$

**Input:** NFA $A = (Q, \Sigma, \delta, q_0, F)$

**Output:** **true** if $\mathcal{L}(A) = \Sigma^*$, **false** otherwise

$\mathcal{Q} \leftarrow \emptyset$;<br>
$\mathcal{W} \leftarrow \{\, \{q_0\} \,\}$<br>
**while** $\mathcal{W} \neq \emptyset$ **do**<br>
    **pick** $Q'$ **from** $\mathcal{W}$<br>
    **if** $Q' \cap F = \emptyset$ **then return false**<br>
    **add** $Q'$ **to** $\mathcal{Q}$<br>
    **for all** $a \in \Sigma$ **do**<br>
        **if** $\mathcal{W} \cup \mathcal{Q}$ contains no $Q'' \subseteq \delta(Q', a)$ **then add** $\delta(Q', a)$ **to** $\mathcal{W}$<br>
**return true**


1
2
3
4
a
b
a, b


1
2
3, 4
1, 3, 4
2, 3
2, 3, 1
2, 3, 4
2, 3, 4, 1
a
b

**Can this approach be correct?**

After all, removing a non-minimal state, we might be preventing the addition of other minimal states in the future!?

## Lemma 34

*Let* $A = (Q, \Sigma, \delta, q_0, F)$ *be an NFA, and let* $B =$ *NFAtoDFA*$(A)$. *After termination of UnivNFA*$(A)$, *the set* $\mathcal{Q}$ *contains all minimal states of* $B$.

## Proof.

Assume the contrary.

Then $B$ has a shortest path $Q_1 \to Q_2 \cdots Q_{n-1} \to Q_n$ such that, after termination,

- $Q_1 \in \mathcal{Q}$, $Q_n \notin \mathcal{Q}$
- $Q_n$ is minimal

Since the path is shortest, $Q_2 \notin \mathcal{Q}$, and so when UnivNFA processes $Q_1$, it does not add $Q_2$. This can only be because UnivNFA already added some $Q_2' \subset Q_2$.

## Proof (cont'd):

But then $B$ has a path $Q'_2 \to Q'_3 \cdots Q'_{n-1} \to Q'_n$ with $Q'_n \subseteq Q_n$. Since $Q_n$ is minimal, $Q'_n = Q_n$ and is minimal.

Thus, the path $Q'_2 \to \cdots \to Q'_n$ satisfies

- $Q'_2 \in \mathcal{Q}$, and
- $Q'_n$ is minimal.

This contradicts our assumption that $Q_1 \to \cdots \to Q_n$ is as short as possible. $\square$

## Inclusion and equality

**Theorem 35**

*The inclusion problem for NFAs is PSPACE-complete.*

**Proof.**

If, given tw o NFAs $A_1$ and $A_2$, we want to test whether $L(A_1) \subseteq L(A_2)$ or, equivalently, $L(A_1) \cap \overline{L(A_2)} = \emptyset$. The negation of the latter can easily be checked (using polynomial space) by guessing a word $w$ (of length at most exponential in the size of $A_1$ and $A_2$) such that $w$ is recognized by $A_1$ but not $A_2$.

PSPACE-hardness on the other hand follows since an NFA $A$ is universal iff $L(A) = \Sigma^*$, *i.e.*, the universality problem reduces to the inclusion problem. $\square$

- Algorithm: use $L_1 \subseteq L_2$ iff $L_1 \cap \overline{L_2} = \emptyset$

- Concatenate four algorithms:
  (1) determinize $A_2$,
  (2) complement the result,
  (3) intersect it with $A_1$, and
  (4) check for emptiness.

- State of (3): pair $(q, Q)$ , where $q \in Q_1$ and $Q \subseteq Q_2$

- Easy optimizations:
  - store only the states of (3), not its transitions;
  - do not perform (1), then (2), then (3): instead, construct directly the states of (3);
  - check (4) while constructing (3);

# Further optimization: subsumption test

## Definition 36

Let $A_1, A_2$ be NFAs, and let $B_2 = \text{NFAtoDFA}(A_2)$. A state $[q_1, Q_2]$ of $[A_1, B_2]$ is minimal if no other state $[q_1', Q_2']$ satisfies $q_1' = q_1$ and $Q_2' \subset Q_2$.

## Lemma 37

*$LL(A_1) \subseteq L(A_2)$ iff every minimal state $[q_1, Q_2]$ of $[A_1, B_2]$ satisfying $q_1 \in F_1$ also satisfies $Q_2 \cap F_2 \neq \emptyset$.*

## Proof.

Since $A_2$ and $B_2$ recognize the same language, $L(A_1) \subseteq L(A_2)$ iff $L(A_1) \cap \overline{L(A_2)} = \emptyset$ iff $L(A_1) \cap \overline{L(B_2)} = \emptyset$ iff $[A_1, B_2]$ has a state $[q_1, Q_2]$ such that $q_1 \in F_1$ and $Q_2 \cap F_2 = \emptyset$. But $[A_1, B_2]$ has some state satisfying this condition iff it has some minimal state satisfying it. □

**Algorithm** `InclNFA`$(A_1, A_2)$:
**Input:** `NFAs` $A_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$, $A_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$
**Output:** **true** `if` $L(A_1) \subseteq L(A_2)$, **false** `otherwise`

$Q := \emptyset$
$W := \{ [q_{01}, \{q_{02}\}] \}$
**while** $W \neq \emptyset$ **do**
  `pick` $[q_1, Q_2]$ `from` $W$
  **if** $q_1 \in F_1$ **and** $Q_2 \cap F_2 = \emptyset$ **then return false fi**
  `add` $[q_1, Q_2]$ `to` $Q$
  **for all** $a \in \Sigma, q_1' \in \delta_1(q_1, a)$ **do**
    $Q_2' := \delta_2(Q_2, a)$
    **if** $W \cup Q$ `contains no` $[q_1'', Q_2'']$ `s.t.` $q_1'' = q_1'$ `and` $Q_2'' \subseteq Q_2'$ **then**
      `add` $[q_1', Q_2']$ `to` $W$
    **fi**
**return true**

- Complexity:
  - Let $A_1, A_2$ be NFAs with $n_1, n_2$ states over an alphabet with $k$ letters.
  - Without the subsumption test:
    - The while-loop is executed at most $n_1 \cdot 2^{n_2}$ times.
    - The for-loop is executed at most $O(k \cdot n_1)$ times.
    - An execution of the for-loop takes $O(n_2^2)$ time.
    - Overall: $O(k \cdot n_1^2 \cdot n_2^2 \cdot 2^{n_2})$ time.
  - With the subsumption case the worst-case complexity is higher. Exercise: give an upper bound.

**Important special case:**

If $A_1$ is an NFA, but $A_2$ (already) is a DFA, then

- complementing $A_2$ is now trivial
- we obtain a running time $O(n_1^2 \cdot n_2)$

**Remark:** To check for equality, we just check inclusion in both directions. To obtain PSPACE-hardness for equality, just observe the universality problem as above.