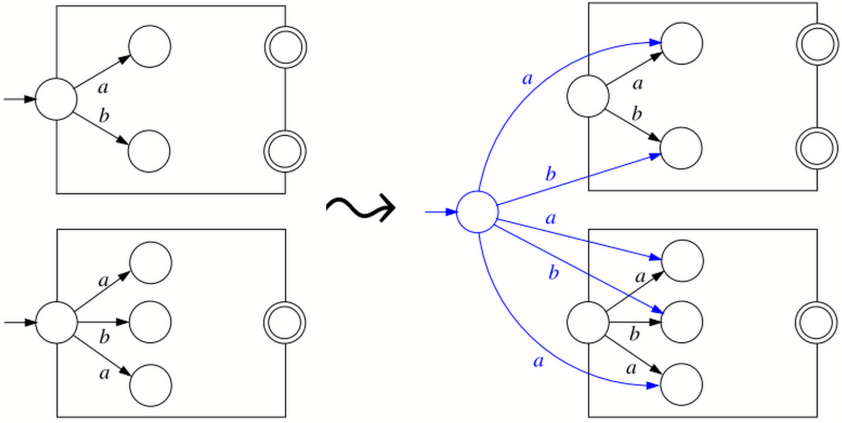
Union



UnionNFA(A_1, A_2)
Input: NFA $A_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$, $A_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$
Output: NFA $A_1 \cup A_2$ with $L(A_1 \cup A_2) = L(A_1) \cup L(A_2)$

```
1  Q ← Q_1 ∪ Q_2 ∪ {q_0}
2  δ ← δ_1 ∪ δ_2
3  F ← F_1 ∪ F_2
4  for all i = 1, 2 do
5     if q_0i ∈ F_i then add q_0 to F
6     for all (q_0i, a, q) ∈ δ_i do
7        add (q_0, a, q) to δ
8     if δ_i^{-1}(q_0i) = ∅ then
9        remove q_0i from Q
10       for all a ∈ Σ, q ∈ δ_i(q_0i, a) do
11          remove (q_0i, a, q) from δ_i
12 return (Q, Σ, δ, q_0, F)
```

Observation:
Clearly, this type of pairing construction does not work for set difference:

SetDiff(A, A) should always produce an NFA recognizing the empty language, but the construction does not work this way!

Emptiness and universality

We observe that an NFA A (in normal form) recognizes the the empty language (*i.e.*, $L(A) = \emptyset$) iff every state of A is non-final.

However, we should also note that the statement

"An NFA is universal iff every state of it is final."

does not hold in general.

In fact, we have

Corollary 30

Emptiness (for DFAs and NFAs) is decidable in linear time.

And ...

Theorem 31

The universality problem for NFAs is PSPACE-complete.

Proof.

We first show that the universality problem is in PSPACE. In fact, we show that it is in NPSPACE and apply Savitch's theorem.

Given an NFA $A = (Q, \Sigma, \delta, q_0, F)$ with $n = |Q|$ states, our algorithm guesses an input for $B = \text{NFAtoDFA}(A)$ leading from $\{q_0\}$ to a non-final state of B, *i.e.*, a set of states of A which are all non-final. If such a run exists, then there is one of length $\leq 2^n$. The algorithm does not store the whole run, only the current state of B, and hence it only needs space linear in n.

Proof (cont'd):

We prove PSPACE-hardness by reduction from the acceptance problem for linearly bounded automata (LBAs). An LBA N is a nondeterministic Turing machine that always halts and only uses the part of the tape containing the input. A configuration of N on an input of length k is encoded as a word of length k. A run of N on an input can be encoded as a word $c_0\#c_1 \ldots \#c_n$, where the c_i's are the encodings of the configurations.

Let Σ be the alphabet used to encode the run of the machine. Given an input x, N accepts if there exists a word w of Σ^* satisfying the following properties:

(a) w has the form $c_0\#c_1 \ldots \#c_n$, where the c_i's are configurations;

(b) c_0 is the initial configuration;

(c) c_n is an accepting configuration; and

(d) for every $0 \leq i \leq n-1$: c_{i+1} is a successor configuration of c_i according to the transition relation of N.

Proof (cont'd):

The reduction shows how to construct in polynomial time, given an LBA N and an input x, an NFA $A(N, x)$ accepting all the words of Σ^* that do not satisfy at least one of the conditions (a)-(d) above. We then have

- If N accepts x, then there is a word $w(N, x)$ encoding an accepting run of N on x, and so $L(A(N, x)) \subseteq \Sigma^* \setminus \{w(N, x)\}$.
- If N does not accept x, then no word encodes an accepting run of N on x, and so $L(A(N, x)) = \Sigma^*$.

Thus, N accepts x if and only if $L(A(N, x)) \neq \Sigma^*$, and we are done. □