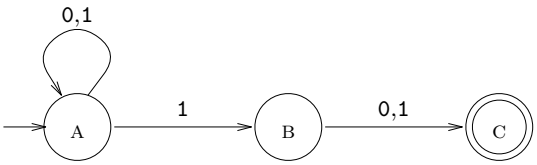
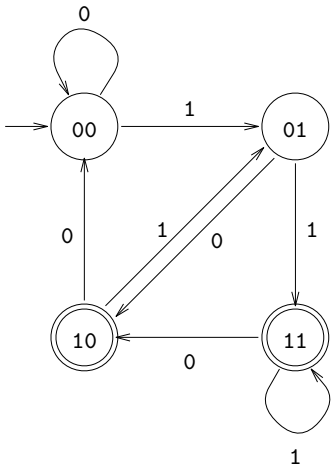
Handout 3: The Subset Construction for Automata

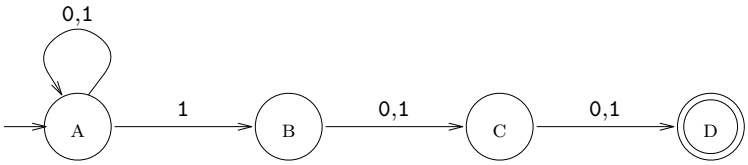
3.1 Example. Check if second to last letter of the input is 1. Nondeterministic state diagram N_2:



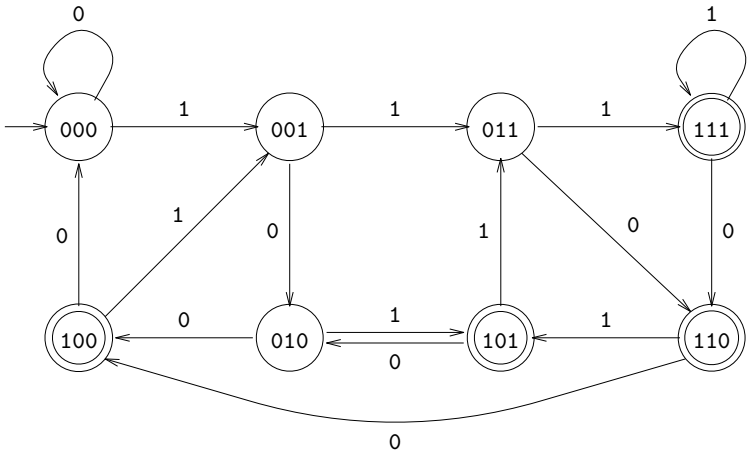
In a deterministic automaton, we need to remember the 2 most recent input symbols. Deterministic state diagram D_2 such that $L(D_2) = L(N_2)$:



3.2 Example. Check if third to last letter of the input is 1. Nondeterministic state diagram N_3:



In a deterministic automaton, we need to remember the 3 most recent input symbols. Deterministic state diagram D_3 such that $L(D_3) = L(N_3)$:

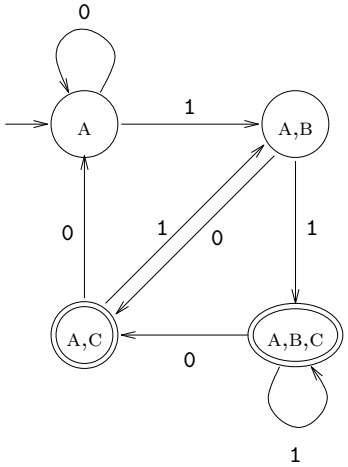


3.3 Nondeterminism is exponentially more succinct than determinism. Check if k-th letter from the end of the input is 1. A nondeterministic finite automaton (NFA) can check this with $k+1$ states; a deterministic finite automaton (DFA) must remember the k most recent input symbols, which requires 2^k states.

3.4 Nondeterminism is no more expressive than determinism. While nondeterminism adds succinctness, it does not add expressiveness: for every NFA N there is an DFA D such that $L(D) = L(N)$. Consider the NFA $N = (Q, \Sigma, \delta, q_0, F)$. We construct an equivalent DFA $D = (Q^d, \Sigma, \delta^d, q_0^d, F^d)$ by recording the set of possible NFA states for each prefix of the input:

$Q^d = \mathcal{P}(Q)$,
$q \in \delta^d(R, a)$ iff $q \in \delta(p, a)$ for some $p \in R$,
$q_0^d = \{q_0\}$,
$R \in F^d$ iff $R \cap F \neq \emptyset$.

Each state of D is a set of states from N. This is called the *subset construction*. If N has n states, then D has 2^n states. Some of these states may be unreachable from the initial state, in which case they can be omitted. Still, as we saw in Section 3.3, in general the exponential cost of determinization cannot be avoided. If we apply the subset construction to the NFA N_2 from Example 3.1, we obtain the following DFA, which is identical to D_2:

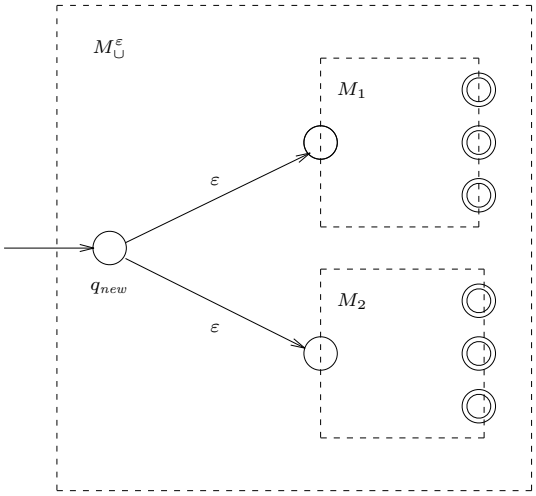


3.4 Epsilon-transitions. It is often convenient to permit an automaton transition without reading an input symbol. Such transitions are called ε-transitions, where ε is the empty word. They often give rise to nondeterministic choice. A finite automaton $M^\varepsilon = (Q, \Sigma, \delta^\varepsilon, q_0, F^\varepsilon)$ with ε-transitions —that is, δ^ε: $Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$— can be converted into an equivalent NFA $M = (Q, \Sigma, \delta, q_0, F)$ without ε-transitions in linear time:

$q \in \delta(p, a)$ iff M^ε has a run of the form $p = p_0 \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} p_n \xrightarrow{a} q$,
$p \in F$ iff M^ε has a run of the form $p = p_0 \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} p_n$ for some $p_n \in F^\varepsilon$.

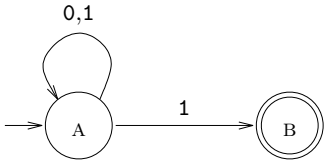
3.5 Intersection and union of nondeterministic automata. The intersection and union of NFAs can be constructed as for DFAs, using the product construction. But there is also a simpler way for building the union. If $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$, then the union can be defined as follows:

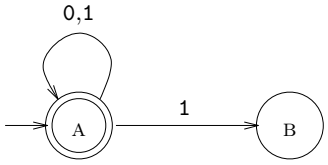
$$M^\varepsilon_\cup = (Q_1 \cup Q_2 \cup \{q_{new}\}, \Sigma, \delta_1 \cup \delta_2 \cup \{(q_{new}, \varepsilon, q_{01}), (q_{new}, \varepsilon, q_{02})\}, q_{new}, F_1 \cup F_2)$$



where $q_{new} \notin (Q_1 \cup Q_2)$. Note that this construction requires $|Q_1| + |Q_2| + 1$ instead of $|Q_1| \cdot |Q_2|$ states. If desired, the two ε-transitions can be removed as in Section 3.4.

3.6 Complementation of deterministic and nondeterministic automata. The complement of a language L over the alphabet Σ is $\overline{L} = \Sigma^* \backslash L$. Consider the two finite automata $M = (Q, \Sigma, \delta, q_0, F)$ and $\overline{M} = (Q, \Sigma, \delta, q_0, Q \backslash F)$. These two automata have the same runs, but an accepting run of M is a rejecting run of $\overline{M}$, and vice versa. If M is deterministic, there is a one-to-one correspondence between input words and runs, and therefore $L(\overline{M}) = \overline{L(M)}$. However, if M is nondeterministic, there is a one-to-many correspondence between input words and runs, and complementing the acceptance condition for runs does not complement the language. For example, both automata





accept the input word 01, because it corresponds to two runs, $\mathrm{A} \xrightarrow{0} \mathrm{A} \xrightarrow{1} \mathrm{B}$ and $\mathrm{A} \xrightarrow{0} \mathrm{A} \xrightarrow{1} \mathrm{A}$, one accepting and the other rejecting (in either automaton). In fact, there is no simpler way for complementing an NFA other than first determinizing it (using the subset construction), and then swapping final and nonfinal states. This, of course, has exponential cost.

3.7 Summary of boolean operations.

	DFAs	NFAs
Intersection of n_1-state automaton with n_2-state automaton	$n_1 \cdot n_2$ states[1]	$n_1 \cdot n_2$ states[1]
Union of n_1-state automaton with n_2-state automaton	$n_1 \cdot n_2$ states[1]	$n_1 + n_2 + 1$ states[3]
Complement of n-state automaton	n states	2^n states[2]

[1] product construction
[2] subset construction
[3] ε-transitions