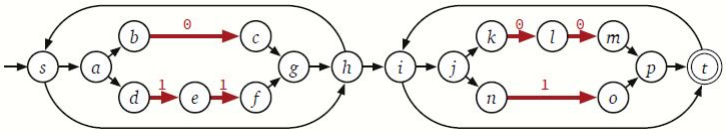
1. For each of the following regular expressions, describe or draw two finite-state machines:

   - An NFA that accepts the same language, constructed from the given regular expression using Thompson's algorithm (described in class and in the notes).
   - An equivalent DFA, constructed from your NFA using the incremental subset algorithm (described in class and in the notes). For each state in your DFA, identify the corresponding subset of states in your NFA. Your DFA should have no unreachable states.

   (a) $(0 + 11)^*(00 + 1)^*$

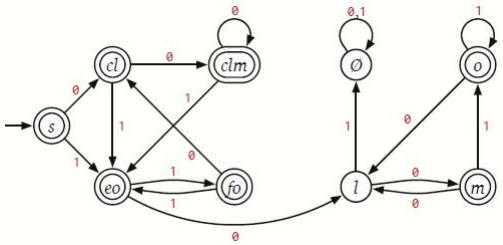
Solution: Thompson's algorithm yields the following 18-state NFA. (Unlabeled arrows indicate ε-transitions.)



The incremental subset algorithm builds the following table.

q'	ε-reach	0	1	A'?
s	$sabdhijknt$	cl	eo	✓
cl	$sabcdghijklnt$	clm	eo	✓
eo	$eijknopt$	l	fo	✓
clm	$sabcdghijklmnt$	clm	eo	✓
l	l	m	$\emptyset$	
fo	$sabdfhijknot$	cl	eo	✓
m	$ijkmnpt$	l	o	✓
o	$ijknopt$	l	o	✓
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	

We obtain the following 9-state DFA:

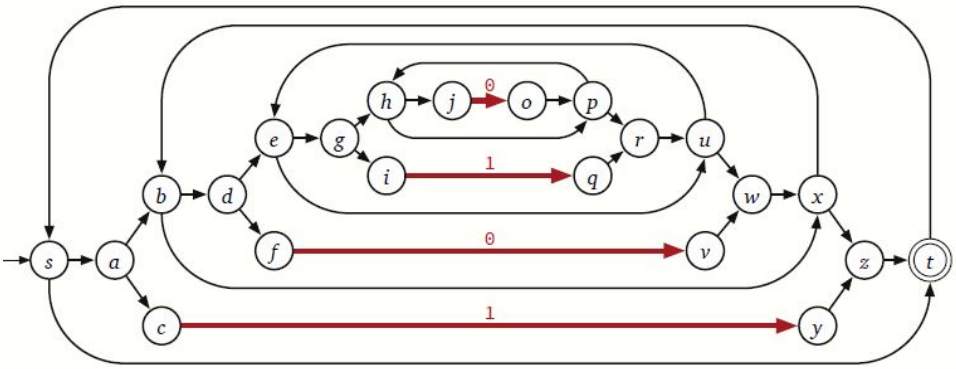


(States s, cl, and clm are equivalent, but simplifying the DFA is beyond the scope of the homework.) ■

Rubric: 5 points = 2½ points for NFA + 2½ points for DFA; see page 3 for details.

(b) $(((\texttt{0}^* + \texttt{1})^* + \texttt{0})^* + \texttt{1})^*$

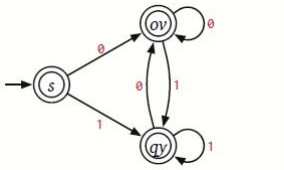
Solution: Thompson's algorithm yields the following 22-state NFA. (As usual, unlabeled arrows indicate ε-transitions.)



The incremental subset algorithm builds the following table. I'm explicitly listing only the *complement* of the ε-reach of each state, both to save space and to protect against errors.

q'	ε-reach	0	1	A'?
s	$Q \setminus oqvy$	ov	qy	✓
ov	$Q \setminus qy$	ov	qy	✓
qy	$Q \setminus ov$	ov	qy	✓

We obtain the following DFA with three(!) states:



(All states are accepting and therefore equivalent, but simplifying the DFA is beyond the scope of the homework.) ■

Rubric: 5 points = 2½ points for NFA + 2½ points for DFA; see next page for details.

Rubric (for each part of problem 2): 5 points =

- 2½ points for Thompson's NFA:
  - No credit unless NFA is actually constructed using Thompson's algorithm.
  - −1 for one small mistake; otherwise, no credit unless the NFA accepts the target language.
  - No penalty for "safe" simplifications, like using a simple path without ε-transitions for individual strings (as shown in the part (a) solution), or generalizing the $+$ gadget to more than two subexpressions, provided these simplifications are applied correctly and consistently.
- 2½ points for incremental-subset DFA:
  - No credit unless the DFA is constructed from the NFA given in part (a) using the incremental subset algorithm, **without further simplifications**.
  - Removing ***all*** ε-transitions from the NFA in part (a) before applying the incremental subset algorithm is fine, as long as it's done systematically (using the algorithm described in the notes) and correctly. This will not change the final DFA.
  - −1 for one small mistake; otherwise, no credit unless the DFA accepts the target language.
  - **Either** the table **or** the drawing (with state labels) is sufficient for full credit; it is not necessary to provide both. The table (even without the ε-reach column) is a complete description of the DFA! **If you provide both, the grader is free to grade *either* the table *or* the drawing.**
  - −1 for a bare drawing without labels correctly indicating the set of NFA-states corresponding to each DFA-state.
  - No further explanation of the DFA is necessary.

2. Let L be any regular language over the alphabet $\Sigma = \{0, 1\}$. Prove that the following languages are also regular.

   (a) $\mathsf{thirds}(L) := \{\mathsf{thirds}(w) | w \in L\}$,
   where $\mathsf{thirds}(w)$ is the subsequence of w containing every third symbol.
   For example, $\mathsf{thirds}(011000110) = 100$.
   (notice, we picked the third, sixth, and ninth symbols in 011000110)

Solution: Let L be an arbitrary regular language, and let $M = (Q, s, A, \delta)$ be a DFA that accepts L. We construct an NFA $M' = (Q', s', A', \delta')$ that accepts $\mathsf{thirds}(L)$ as follows:

$$Q' = Q$$
$$s' = s$$
$$A' = A \cup \left\{ q \in Q \middle| \Big(\delta(q,a) \cup \delta(\delta(q,a),b)\Big) \cap A \neq \varnothing \text{ for some } a \in \Sigma, b \in \Sigma \right\}$$
$$\delta'(q,a) = \bigcup_{b \in \Sigma, c \in \Sigma} \{\delta(\delta(\delta(q,c),b),a)\}$$

M' reads the input string $\mathsf{thirds}(w)$ and simulates M running on string w, while non-deterministically guessing the missing symbols in w.

- When M' reads symbol a from $\mathsf{thirds}(w)$, it guesses symbols $b, c \in \Sigma$ and simulates M reading cba from w.
- When M' finishes $\mathsf{thirds}(w)$, it guesses whether w has length $3k$ or $3k+1$ or $3k+2$ for non-negative integer k, and if needed, it guesses the last one or two characters of w.

■

Rubric: 5 points: standard language transformation rubric (scaled). This is not the only correct solution.

(b) $\mathsf{thirds}^{-1}(L) := \{w \in \Sigma^* | \mathsf{thirds}(w) \in L\}$.

Solution: Let L be an arbitrary regular language, and let $M = (Q, s, A, \delta)$ be a DFA that accepts L. We construct a **DFA** $M' = (Q', s', A', \delta')$ that accepts $thirds^{-1}(L)$ as follows:

$$Q' = Q \times \{0, 1, 2\}$$
$$s' = (s, 0)$$
$$A' = A \times \{0, 1, 2\}$$
$$\delta'((q, i), a) = (q, i + 1) \text{ for } i \in \{0, 1\}$$
$$\delta'((q, 2), a) = (\delta(q, a), 0)$$

M' reads its input string w and simulates M running on $\mathsf{thirds}(w)$.

- State $(q, 0)$ means M' has just read a third symbol in w, so M should ignore the next symbol.
- State $(q, 1)$ means M' has just read symbol $3k + 1$ in w for non-negative integer k, so M should ignore the next symbol.
- State $(q, 2)$ means M' has just read symbol $3k + 2$ in w for non-negative integer k, so M should read the next symbol.

■

Rubric: 5 points: standard language transformation rubric (scaled). This is not the only correct solution.