**CS 164 Programming Languages and Compilers** **Spring 2019**

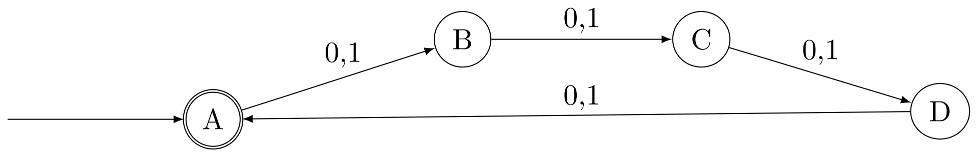
# Discussion Week of 2/4: Finite Automata

Instructor: Paul N. Hilfinger GSIs: Nikhil Athreya, Vivant Sakore

## 1 DFA and NFA

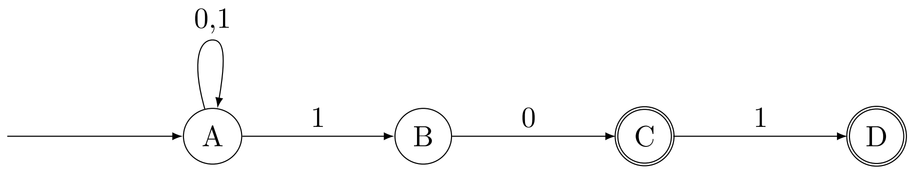
Finite state automata (FSA) are abstract machines that feature states and guarded transitions from one state to another. An FSA can only be in one state a time, and its total number of states is *finite*. The machine takes transitions in response to inputs it receives sequentially; if an input matches the guard of a transition that departs from the current state that transition is said to be *enabled*; only enabled transitions can be taken. FSA that have an accepting state provide the machinery to determine whether an input string is in a regular language. If no transitions are enabled by a given input then the entire input string gets rejected. If all inputs are processed and the FSA is in an accepting state then the entire input string is accepted. Deterministic finite state automata (DFA) can only have one enabled transition at a time while a non-deterministic finite state automata (NFA) can have multiple.

1. What language is accepted by the following DFA?



**Answer:** Binary strings of length 4 or multiples of 4, and the empty string.

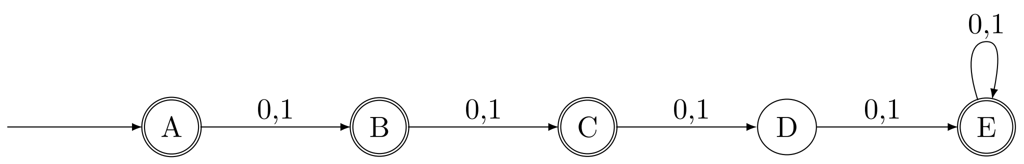
2. What language is accepted by the following NFA?



**Answer:** Binary strings ending in 10 or 101.

3. Construct a DFA that accepts binary strings of any length, except 3.

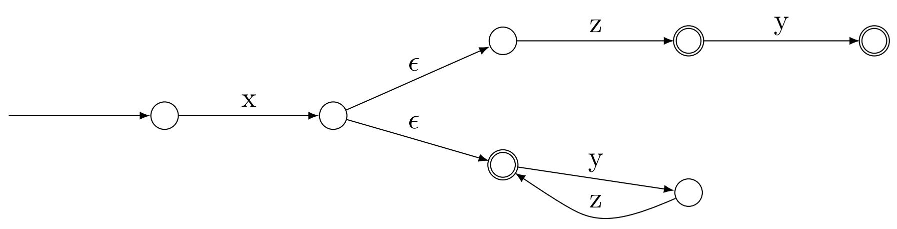
**Answer:**



4. Construct a NFA that accepts:

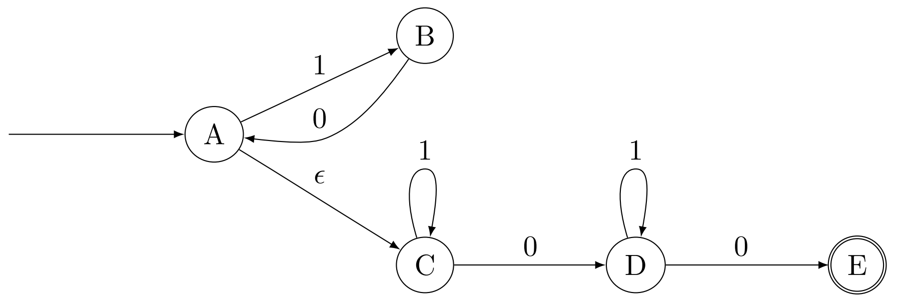
   (a) The language denoted by `x(zy?|(yz)*)`.

   **Answer:**

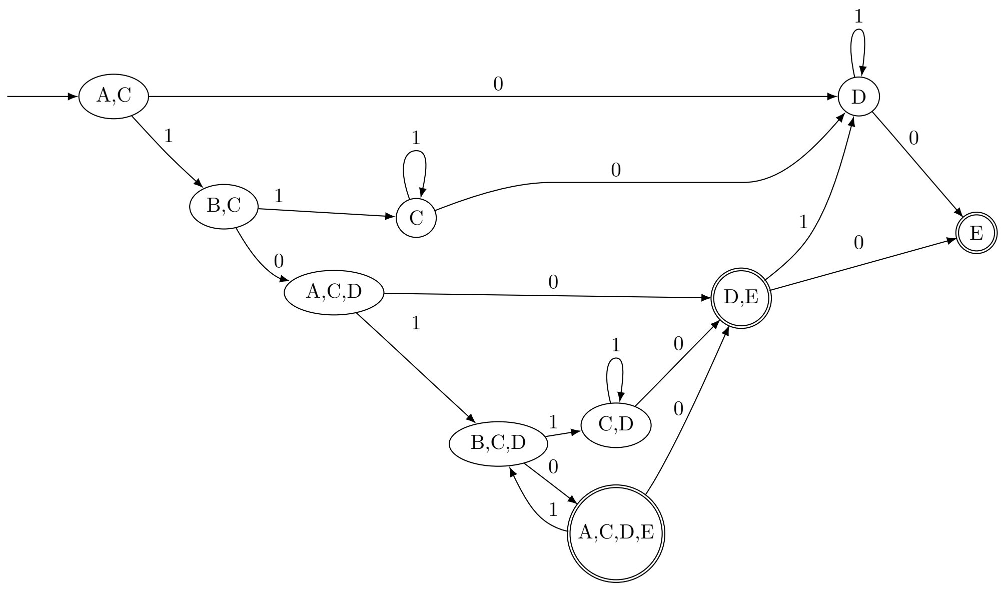


   (b) The language denoted by `(10)*1*01*0`.

   **Answer:**



5. Convert the NFA in Exercise 4b to a DFA. **Answer:**



## 2 Pumping Lemma

**Pumping Lemma**: For any DFA (or NFA or regular expression) that accepts an infinite number of strings, there is some minimum length, $M$, such that any string with length greater than or equal to $M$ that the machine accepts must have the form $uxv$, where $u$, $x$, and $v$ are strings, $x$ is not empty, the length of $ux$ is $\leq M$, and the machine accepts all strings of the form $ux^nv$, for all $n \geq 0$.

6. Let $A = \{1^j z | z \in \{0,1\}^*$ and $z$ contains at most $j$ 1's, for any $j \geq 1\}$. Prove, by the pumping lemma, that $A$ is not regular.

   **Proof**: Assume, for the sake of contradiction, that $A$ is regular.

   Let $M$ be the minimum length as in the pumping lemma, for the regular language $A$. Select $s = 1^M 0 1^M$, which is accepted by $A$. The pumping lemma says that if $A$ is regular, there is a decomposition $s = uxv$ such that $|x| > 0, |ux| \leq M$, and $ux^nv \in A$ for all $n \geq 0$. Since $|ux| \leq M$, the substring $ux$ must be entirely contained within the initial $1^M$ substring of $s$. Thus, $u = 1^p$ and $x = 1^q$ for some integers $p$ and $q$ such that $p \geq 0$, $q \geq 1$, (since $q = |x| > 0$), and $p + q \leq M$.

   Then $s = uxv = 1^p 1^q 1^{M-p-q} 0 1^M$. Now $ux^nv$ is supposed to be in $A$ for any $n \geq 0$, so in particular $ux^0v = uv$ should be in $A$.

   But $uv = 1^p 1^{M-p-q} 0 1^M = 1^{M-q} 0 1^M$, and $q \geq 1$, so $M - q < M$. This word begins with $M - q$ ones followed by a zero, so if we try to match it up with the form $1^j z$ in the definition of $A$, the largest value of $j$ that it can match is $j = M - q$, which is less than $M$. But then $z$ has to be $01^M$, which has more than $j$ ones, and thus $uv$ can't be in $A$ after all. This is a contradiction, and consequently $A$ is not regular. ■