# CSC236 Week 07: Autoamata and Languages

Hisbaan Noorani

October 21 – October 27, 2021

## Contents



## 1 Example — an odd machine

A machine that accepts strings over $\{0, 1\}$ with an odd number of 0s.

We will formally prove that this description can be represented by:

$$\delta^*(E, s) = \begin{cases} E & \text{only if } s \text{ has even number of 0s} \\ O & \text{only if } s \text{ has odd number of 0s} \end{cases}$$

Define $\Sigma^*$ as the smallest set of strings over $\Sigma$ such that:

- $\varepsilon \in \Sigma^*$
- $s \in \Sigma^* \implies s0, s1 \in \Sigma^*$

Define $P(s)$ as $\delta^*(E, s)$ correctly defines the machine.

*Proof:* We will show that $\forall s \in \Sigma^*, P(s)$.

- Base Case: The following implications hold vacuously:

$$\delta^*(E, \varepsilon) = E \implies \varepsilon \text{ has an even number of 0s}$$

$$\delta^*(E, \varepsilon) = O \implies \varepsilon \text{ has an odd number of 0s}$$

And thus for all members of the basis, $P(s)$.

- Inductive step: Let $s \in \Sigma^*$ and assume $P(s)$. We want to show that $P(s0)$ and $P(s1)$ hold.

$P(s0)$: If $\delta^*(E, s) = E$

$$\begin{aligned} \delta^*(E, s0) &= \delta(\delta^*(E, s), 0) \\ &= \delta(E, 0) && \text{(by the current case)} \\ &= O \end{aligned}$$

So $s0$ has an odd number of 0s and $P(s0)$ follows in this case.
If $\delta^*(E, s) = O$

$$\begin{aligned} \delta^*(E, s0) &= \delta(\delta^*(E, s), 0) \\ &= \delta(O, 0) && \text{(by the current case)} \\ &= E \end{aligned}$$

So $s0$ has an even number of 0s and $P(s0)$ follows in this case.
So $P(s0)$ holds.

$P(s1)$: If $\delta^*(E, s) = E$

$$\begin{aligned} \delta^*(E, s1) &= \delta(\delta^*(E, s), 1) \\ &= \delta(E, 1) && \text{(by the current case)} \\ &= E \end{aligned}$$

So $s1$ has an even number of 0s and $P(s1)$ follows in this case.
If $\delta^*(E, s) = O$

$$\begin{aligned} \delta^*(E, s1) &= \delta(\delta^*(E, s), 1) \\ &= \delta(O, 1) && \text{(by the current case)} \\ &= O \end{aligned}$$

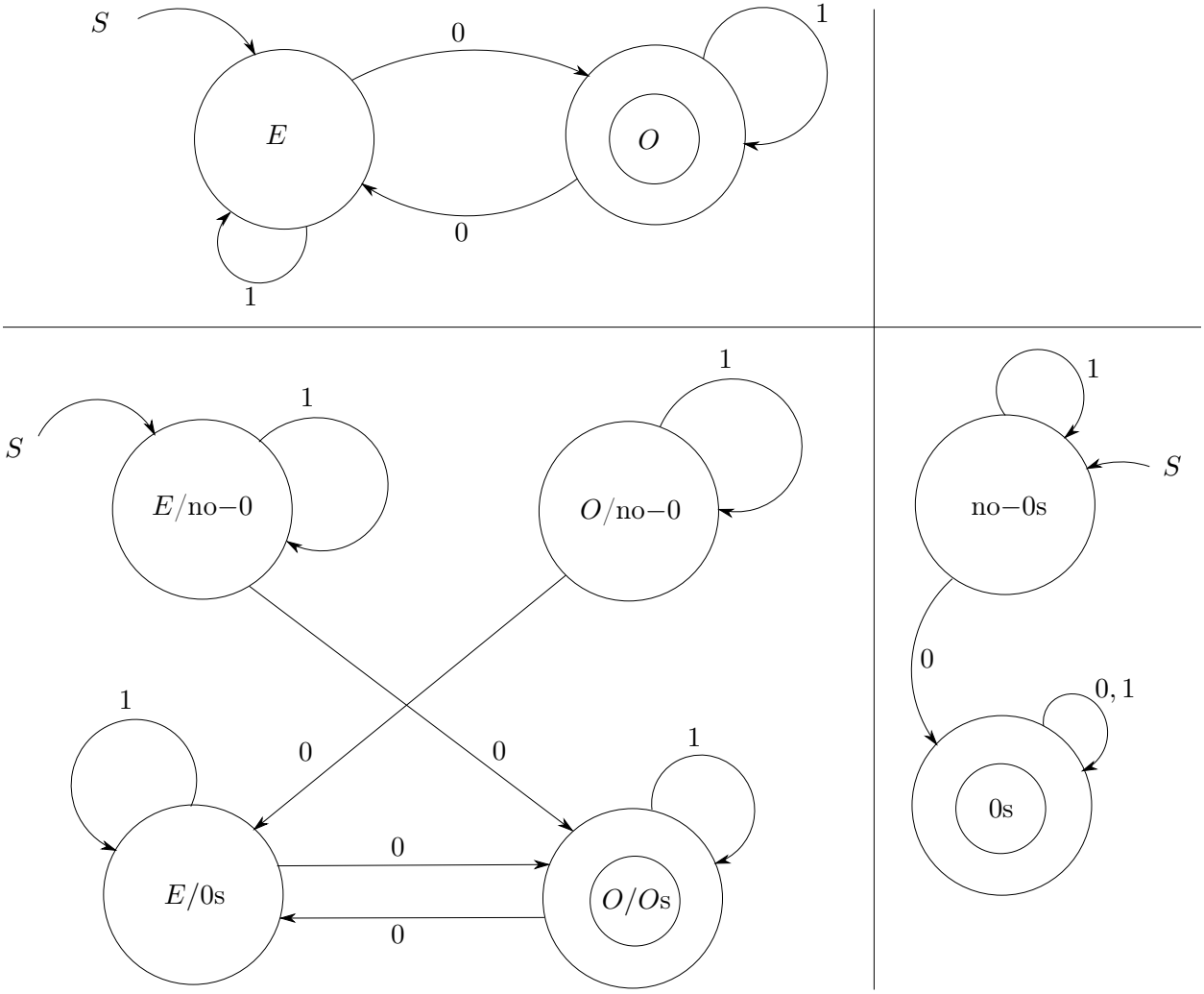
So $s1$ has an odd number of 0s and $P(s1)$ follows in this case.
So $P(s1)$ holds.

Since $P(s0)$ and $P(s1)$ both hold, we have shown that $s \in \Sigma^* \wedge P(s) \implies P(s0) \wedge P(s1)$.

So $\forall s \in \Sigma^*, P(s)$, as needed. ■

## 2 More odd/even: intersection

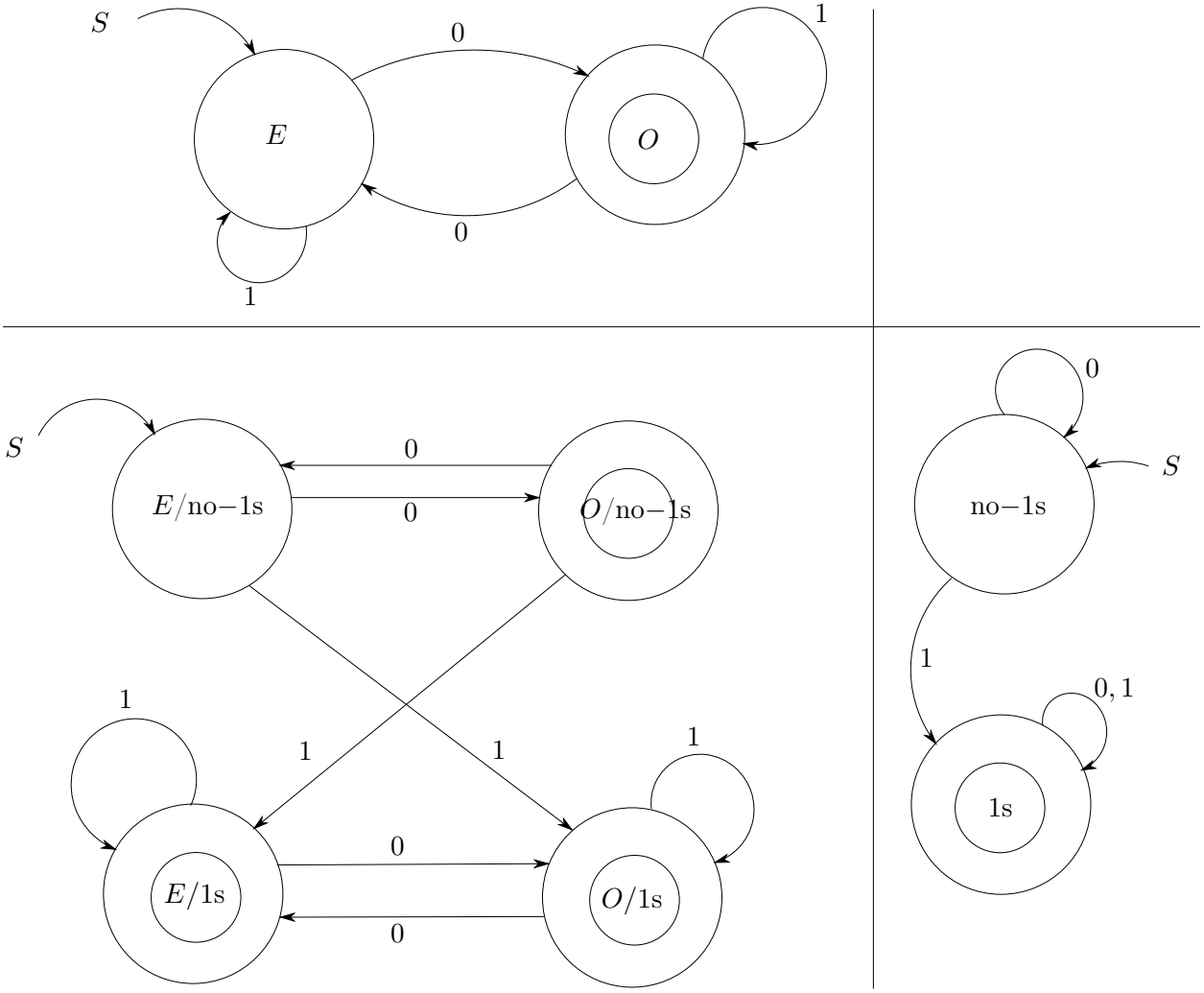
$L$ is the langauge of binary strings with an odd number of 0s, and at least one 0. We will devise a machine for $L$ using product construction.




If there is an odd number of 0s, we don't need to check if we have 0s or not. This means that we do not need to check for the upper right hand state. We could simply leave this state out since there are no arrows going into it.

## 3 More odd/even: union

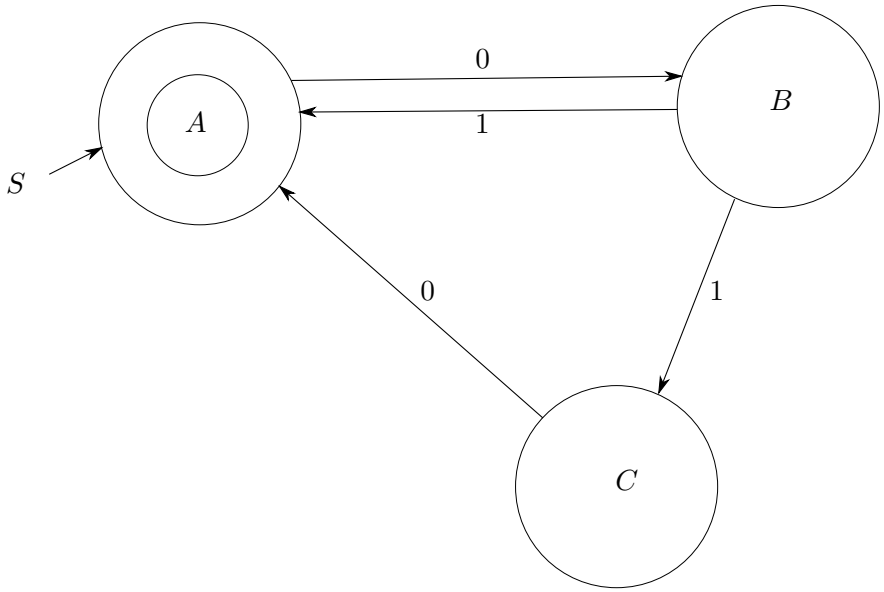
$L$ is the langauge of binary strings with an odd number of 0s, or at least one 1. We will devise a machine for $L$ using product construction.




## 4 Non-deterministic FSA (NFSA) example

FSA that accepts $L((010 + 01)^*)$. With NFSA, we allow more than one transition from a state for the same character. There is no "definite" transition for a given character, there can be different "paths" that a string would take since there can be more than one transition for a given character.

- Accepts: $\varepsilon, 01001, 0101, 01, \ldots$
- Rejects: $1, 110, 10, 01011, \ldots$




A string is accepted if there is some path that accepts the string (goes from the starting state to the accepted state).

## 5 NFSAs are real... you can always convert them to DFSAs

Use subset construction, notes page 219 if $\Sigma = \{0, 1\}$, the construction is, roughly:

- Start at the start state combined with any states reachable from start with $\varepsilon$-transitions.
- If there are ant 1-transition from this new combined start state, combine them into a new state.
- There are any 0-translations from this new combined start, combine them into a new state.
- Repeat for every state reachable from the start.

This system lets us deal with the creation of deterministic finite state automata for regular expressions much more easily by making it a non-deterministic finite state automata and then converting it into a deterministic one.

## 6 Deterministic FSA (DFSA) from NFSA

We will follow the algorithm to construct a DFSA from the previous NFSA.