Lecture 11

Regular Expressions and Non-regular Languages

07-30

Recap

Regular Languages

The following are equivalent

- A is regular
- There is a DFA M such that $L(M) = A$
- There is a NFA N such that $L(N) = A$

Closure

If A, B are regular, so are

- $\overline{A}$
- $A \cup B$
- $A \cap B$
- AB
- A^n
- A^*

Regular Expressions

Let Σ be an alphabet. Define the set of regular expressions $\mathcal{R}_\Sigma$ recursively as follows.

$\mathcal{R}_\Sigma$ is the smallest set such that

- $\emptyset \in \mathcal{R}_\Sigma$
- $\epsilon \in \mathcal{R}_\Sigma$
- $a \in \mathcal{R}_\Sigma$ for each $a \in \Sigma$
- $R \in \mathcal{R}_\Sigma \implies (R)^* \in \mathcal{R}_\Sigma$
- $R_1, R_2 \in \mathcal{R}_\Sigma \implies (R_1 R_2) \in \mathcal{R}_\Sigma$
- $R_1, R_2 \in \mathcal{R}_\Sigma \implies (R_1 | R_2) \in \mathcal{R}_\Sigma$

Regular Expressions

The language if a regular expression R, denoted $L(R)$ is the set of strings that R matches. Formally,

- $L(\emptyset) = \emptyset$
- $L(\epsilon) = \{\epsilon\}$
- $L(a) = \{a\}$, for $a \in \Sigma$
- $L(R^*) = L(R)^*$ for $R \in \mathcal{R}_\Sigma^*$
- $L(R_1 R_2) = L(R_1)L(R_2)$ for $R_1, R_2 \in \mathcal{R}_\Sigma$
- $L(R_1 | R_2) = L(R_1) \cup L(R_2)$ for $R_1, R_2 \in \mathcal{R}_\Sigma$

Today

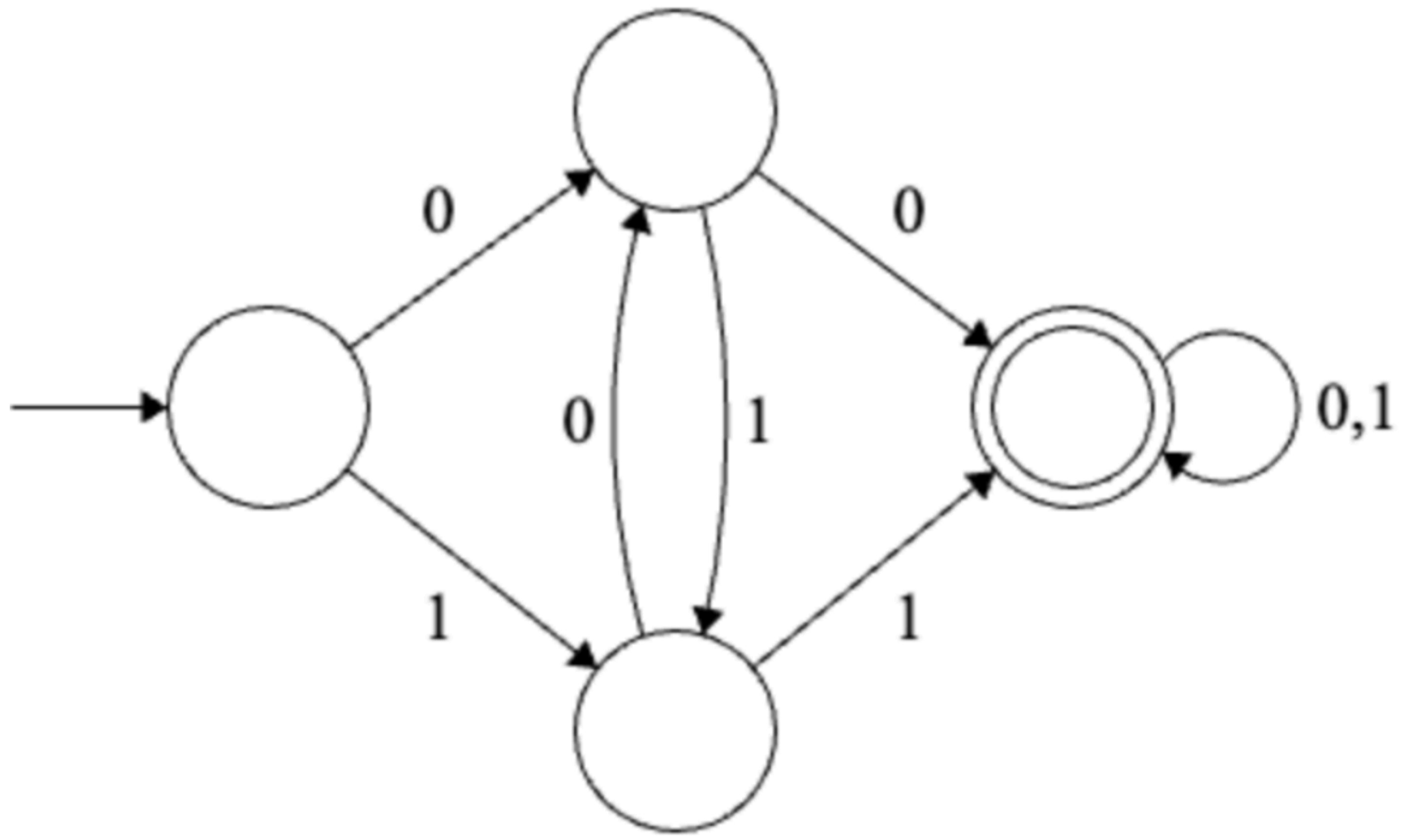
- NFA $\Longleftrightarrow$ Regular Expressions (!!!!)
- Non-regular languages (!!!!)

Equivalence of NFAs and Regular Expressions

Regex -> NFA

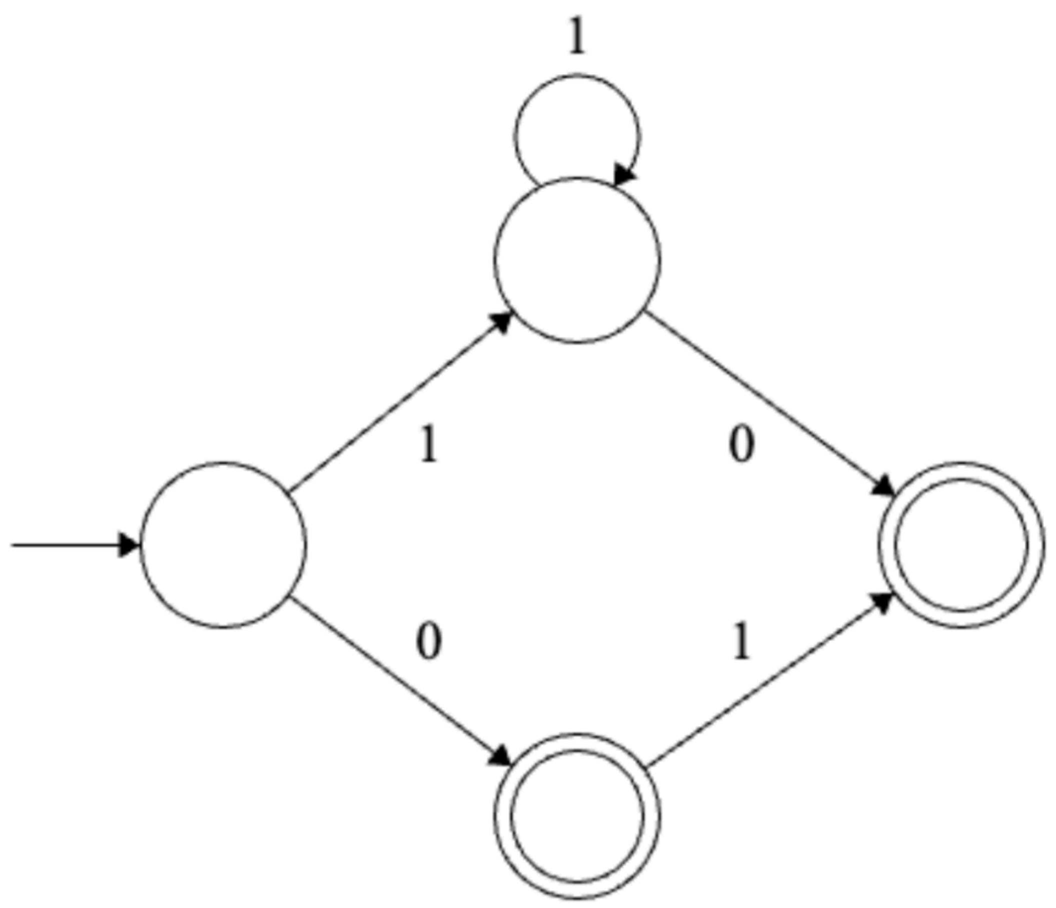
Equivalence of NFAs and Regular Expressions

NFA -> Regular Expression



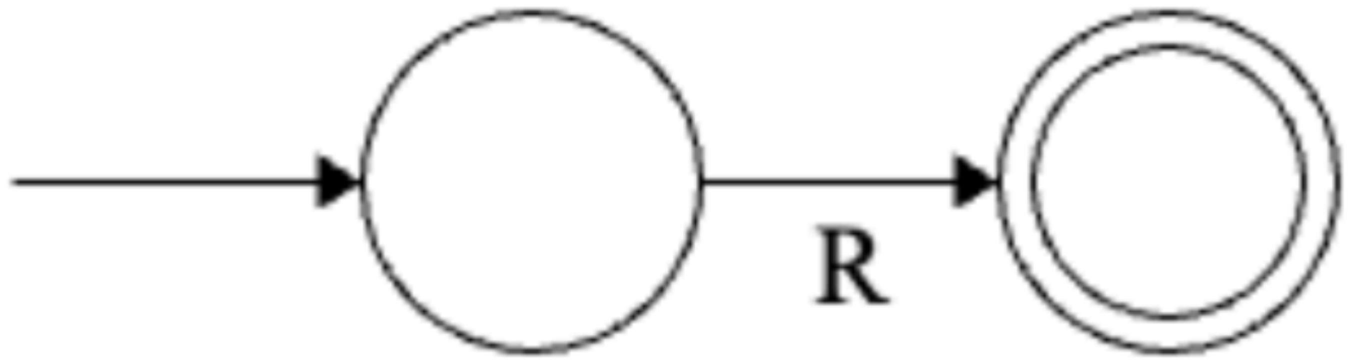
Equivalence of NFAs and Regular Expressions

NFA -> Regular Expression



Sketch of Formal Proof

- Alter the NFA so there's just one accepting state (using ϵ transitions).
- Iteratively rip out states, replacing transitions with regular expressions until you have something that looks like



R is the equivalent regular expression.

For two states q_1, q_2 with a transition between them, let $f(q_1, q_2)$ be the regular expression labelling the transition.

Here are the steps to rip out a state q.

1. **Remove the loop**: If there is a self loop on state q, for each state s with a transition into q, update the transition $f(s, q) = f(s, q)f(q, q)^*$. For each state s' with a transition out of q, update the transition $f(q, s') = f(q, q)^* f(q, s')$

2. **Bypass** q: for each path (s, q, t) of length 2 through q, update $f(s, t) = f(s, t)|f(s, q)f(q, t)$. Note that it is possible that $s = t$, in which case this step adds a loop.

3. Remove q.

Regular Languages

The following are equivalent

- A is regular
- There is a DFA M such that $L(M) = A$
- There is a NFA N such that $L(N) = A$
- **There is a regular expression** R **such that** $L(R) = A$

Showing a language is regular

- Find either a DFA, NFA, or Regular Expression for the language!

-

How to choose

- I typically use regular expressions for languages that seem to require some form of 'matching'. For example *contains 121* as a substring, or *ends with 11*. Regular expressions are typically faster to find and write out in an exam setting.
- I'll use NFAs when I can't easily figure out a regular expression for something. These are usually languages for which memory seems to be useful like the $\mathrm{Dogwalk}$ example from hw.
- Stuff involving negations also seems easier to do with NFAs than with regular expressions. For example, *contains the substring* 011 is easy with regular expression, but *doesn't contain the substring* 011 is a bit more complicated.

Non-Regular Languages

- Are all languages regular?

Key Intuition

- Regular $\iff$ Computable with “finite memory”

Example

- Let $X = \{a^n b^n : n \in \mathbb{N}\}$. Claim: X is not regular
- Why is this the case, using the intuition from the previous slide?

Proof

Same State, Same Fate

- If two strings x and y reached the same state, then no matter what string w comes after, xw and yw will end up in the same state and hence will both be accepted or both be rejected

- Equivalently, different fates $\rightarrow$ different states

- X has infinitely many strings that have different fates, hence, there must be infinitely many states!

Distinguishable

Myhill-Nerode Theorem

Let A be a language over Σ. Suppose there exists a set $S \subseteq \Sigma^*$ with the following properties

- (Infinite). S is infinite
- (Pairwise distinguishable). $\forall x, y \in S$, with $x \neq y$. x, and y are distinguishable relative to A.

Then A is not regular.

Using the Myhill-Nerode Theorem

- By the Myhill-Nerode Theorem, to show a language A is not regular, it suffices to find a set $S \subset \Sigma^*$ such that S is infinite, and pairwise distinguishable relative to A.

Example

Showing X is not regular using the Myhill-Nerode Theorem.