# Lecture 10

## Regular Languages

2025-07-23

# Review

## DFAs

# Review

## NFAs

# Regular Languages

- A language $A \subseteq \Sigma^*$ is regular if and only if there exists some DFA $M$, such that $L(M) = A$

# Complement

- **Claim.** If $A$ is regular, then so is $\overline{A}$

# Union

- **Claim.** If $A, B$ are regular, then so is $A \cup B$

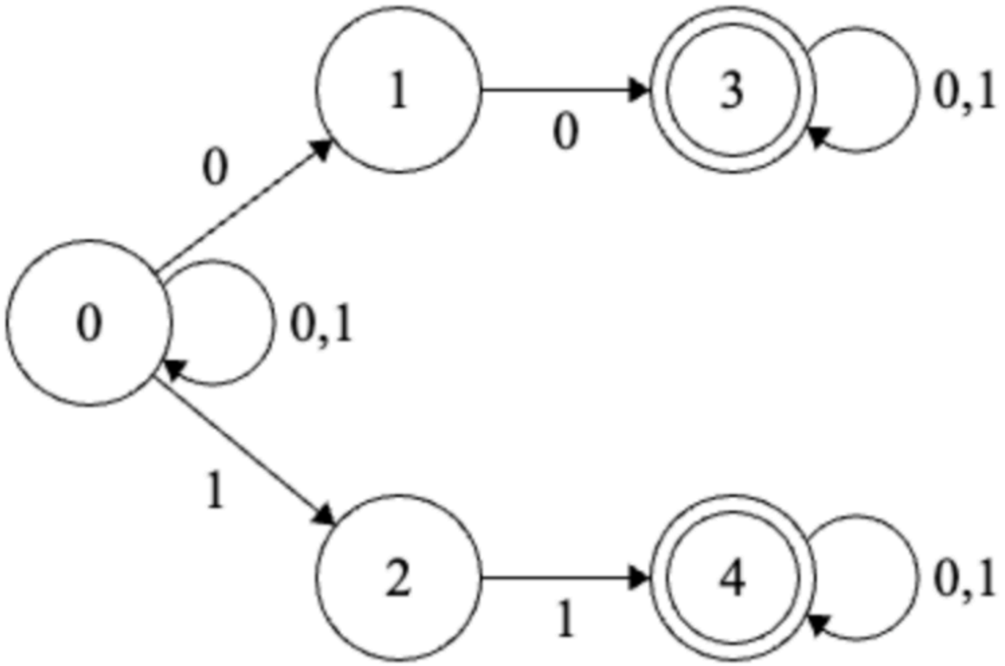
# Closure

## If $A, B$ are regular, then so are…

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

# Concatenation: $AB$

# Equivalence of NFAs and DFAs


0
1
2
3
4
0
1
0
1
0,1
0,1
0,1

# Regular Languages

$AB$

$A^n$

$A*$

# Regular Expressions

- Fix an alphabet $\Sigma$. Then $R_\Sigma$, the set of regular expressions over the alphabet $\Sigma$ is defined recursively/inductively as follows.
- Base cases: $B = \Sigma \cup \{\emptyset, \epsilon\}$
- Functions:
  - $f_1(R) = (R*)$
  - $f_2(R, S) = (RS)$
  - $f_3(R, S) = (R \mid S)$
- $R_\Sigma$ is the set generated by $\{f_1, f_2, f_3\}$ from the base cases $B$.

# Precedence

- * then concatenation then |

# Language of a Regular Expression

# Examples

- $((0|1)(0|1)(0|1))^*$
- $(1^*01^*0)^*$
- $(0|1)^*1(0|1)(0|1)$
- $(0|1)^*010(0|1)^*$
- $(0|\epsilon)1^*$

# Shorthand

If $R$ is a regex, and $m \in \mathbb{N}$, then

- $R^m$ means $m$ copies of $R$.
- $R^+ = RR^*$, i.e. at least one copy of $R$.
- $R^{m+} = R^m R^*$ means at least $m$ copies of $R$.

If $S = \{s_1, s_2, \ldots, s_n\}$ is a finite set of strings, then $S$ is shorthand for $s_1|s_2|\ldots|s_n$. A common one is to use the alphabet, $\Sigma$.

# More Examples

Write regular expressions for the following languages

- Starts with 010.
- The second and the second last letter of w are the same.
- Contains 110.
- 1s always occur in pairs.
- Doesn’t contain more than four 1s in a row.

# Equivalence of NFAs and Regex

# Regex $\Longrightarrow$ $NFA$

# Regular Languages

- The following are equivalent
  - $A$ is regular
  - There exists a DFA $M$ such that $L(M) = A$
  - There exists a NFA $N$ such that $L(N) = A$
  - There exists a regular expression $R$ such that $L(R) = A$

# 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.