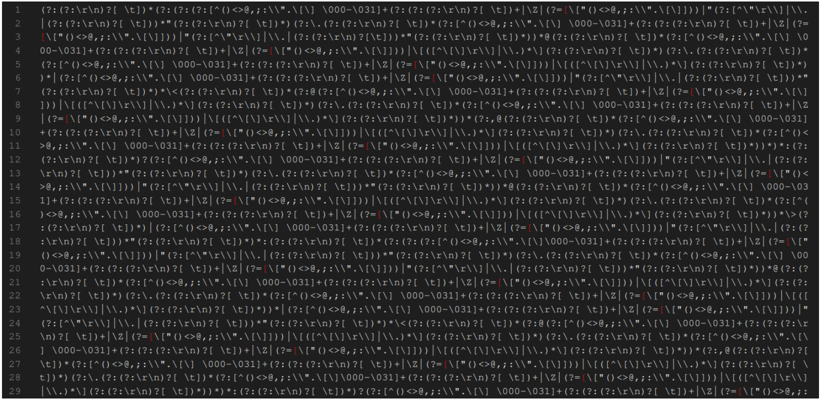
Week 3: Regular expressions, languages, and the pumping lemma for regular languages



Textbook: 1.3 and 1.4

Regular languages

Def: Regular languages. A language is said to be "regular" iff there exists a DFA recognizing it.

Corollary: Since DFA are closed under the regular operations, so to are regular languages.

Corollary: Since DFA and NFA are equivalent, a language is regular iff there exists an NFA recognizing it.

Corollary: All finite sets are regular. Since the language consisting of exactly one string is trivially decidable by a DFA and regular languages are closed under union, any finite set of strings is regular.

Regular expressions

- Invented by Kleene (of the Kleene star, Kleene plus, and Kleene recursion theorem)
- Formal notation for expressing DFAs
- Commonly used in non-theoretical computer science

Formal definition of regular expressions

Def: Regular expressions. We say R is a regular expression on alphabet Σ if R is:

1) a character $\in \Sigma$
2) the **empty string** ϵ
3) the **empty set** $\emptyset$
4) $(R_1 \cup R_2)$ where R_1, R_2 are regular expressions
5) $(R_1 \circ R_2)$ where R_1, R_2 are regular expressions
6) $(R_1)^*$ (zero or more repetitions of R_1) where R_1 is a regular expression

RegEx shorthands

- For convenience, $A \circ B$ is written AB
- A^+ is shorthand for $A \circ A^*$
- **Informally,** $A \cup B$ is given by $(A|B)$. This is used in real life, but not here
- Σ is usually inferred

Thm: Regular expressions and NFA are equivalent. This will be informally proved after this slide.

Corollary: A language is regular iff there exists a regular expression recognizing it.

RegEx-NFA equivalence pt. 1

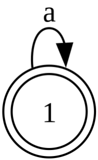


Figure 1: a for some $a \in \Sigma$

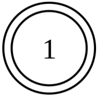


Figure 2: ϵ

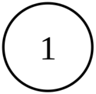


Figure 3: $\emptyset$

RegEx-NFA equivalence pt. 2

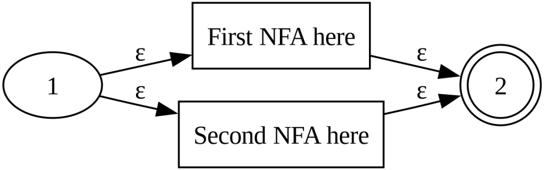


Figure 4: $(R_1 \cup R_2)$

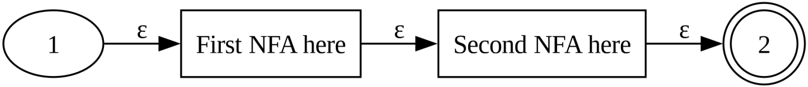
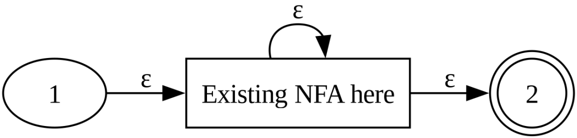


Figure 5: $(R_1 \circ R_2)$



Generalized Nondeterministic Finite Automata (GNFAs) pt. 1

- We can use the previous diagrams to convert REs to NFAs (and thus to DFAs)
- **How do we convert NFAs to REs?**

Def: Generalized nondeterministic finite automata (GNFAs). A GNFA is a NFA where edges are allowed to be any RE, rather than a character $a \in \Sigma$ for alphabet Σ. For convenience, we make the following assumptions:

1) The start state has transition arrows going to every other state but no arrows coming in from any other state
2) There is only a single accept state, and it has arrows coming in from every other state but no arrows going to any other state
3) The accept state is not the start state
4) Except for the start and accept states, one arrow goes from every state to every other state and also from each state to itself

GNFAs pt. 2

- It is trivial to show that any NFA can be converted to an equivalent one in this form

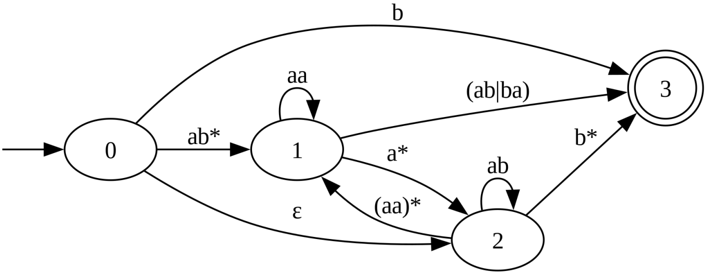


Figure 7: A generalized nondeterministic finite automaton

Non-regular languages

The pumping lemma for regular languages

Next up: Context-free grammars