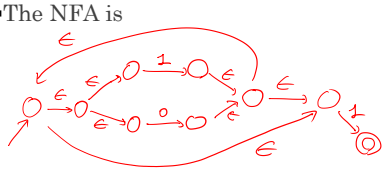
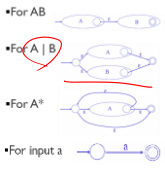
Regular Expressions to NFA

- Consider the regular expression
  - (1 | 0)*1
- The NFA is

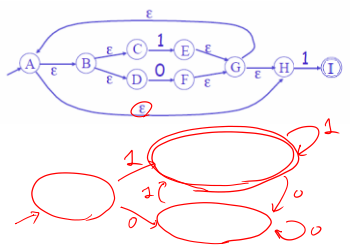




NFA to DFA

- Simulate the NFA
- Each state of DFA
  - a non-empty subset of states of the NFA
- Start state
  - the set of NFA states reachable through ε-moves from NFA start state
- Add a transition S $\rightarrow^{a}$ S' to DFA iff
  - S' is the set of NFA states reachable from any state in S after seeing the input a, considering ε-moves as well

NFA to DFA



Start state
the set of NFA states reachable through ε-moves from NFA start state
Add a transition S $\rightarrow^{a}$ S' to DFA iff
S' is the set of NFA states reachable from any state in S after seeing the input a, considering ε-moves as well

Practice

- Convert the regular expression to DFA
  - z (x | y)*

Regular Language Wrap-up

- Regular expressions generate a regular language.
  - The set of all strings in the regular language can be described by regular expression.
- Finite Automata recognize the regular language.
- Writing a pure DFA as a set of nested case statements is a useful programming technique.
  - We can build a scanner: tokenizing source

Limitation of Regular Expression

- How to write a regular expression for matching parenthesis?
  - L(R) = {ε, (), (()), ((())), …}
- Regular languages cannot express languages with properties that we care about.

Context-Free Grammars

The Java® Language Specification

Java SE 9 Edition

2017-08-07

Legal Notice

Table of Contents

1. Introduction
   - 1.1. Organization of the Specification
   - 1.2. Example Programs
   - 1.3. Notation
   - 1.4. Relationship to Predefined Classes and Interfaces
   - 1.5. Feedback
   - 1.6. References
2. Grammars
   - 2.1. Context-Free Grammars
   - 2.2. The Lexical Grammar
   - 2.3. The Syntactic Grammar
   - 2.4. Grammar Notation

Context-Free Grammars

- The notation for context-free grammars (CFGs) is also called Backus-Naur Form (BNF).
- A CFG consists of
  - A set of terminals: t
  - A set of non-terminals: N
  - A start symbol: S
  - A set of productions: $X \longrightarrow Y$
- CFGs are a natural notation for the recursive structure.
- CFG is a generator for a context-free language.

Context-Free Grammars

- Expression grammar with precedence and associativity

$$expr \longrightarrow \texttt{id} \mid \texttt{number} \mid \texttt{-}\ expr \mid \texttt{(}\ expr\ \texttt{)} \mid expr\ op\ expr$$

$$op \longrightarrow \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{/}$$