Compiler Construction (CS-636)

Sadaf Manzoor
UIIT, Rawalpindi

Outline

1. RE to NFA Conversion
2. The Parsing Process
3. Context-Free Grammars
4. Summary

Context-Free Grammars and Parsing

Lecture: 7 & 8

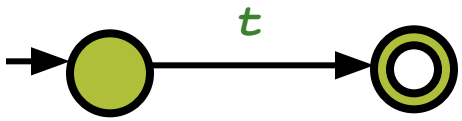
RE to NFA Conversion

- A regular expression can be converted to NFA that accepts that same language as of RE
- The NFA can also be converted into equivalent DFA (it is beyond the scope of our course)
- The input of the process is regular expression “r” over alphabet ∑ and the output is NFA “N” accepting L(r)

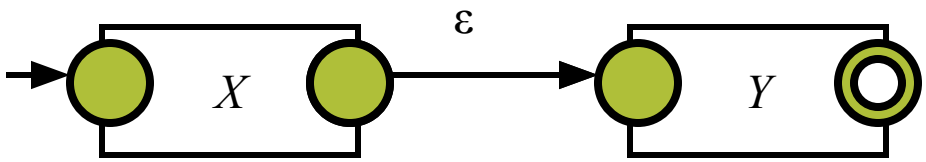
RE to NFA Conversion (Continue…)

RE: ε
FA:

RE: `t`
FA:



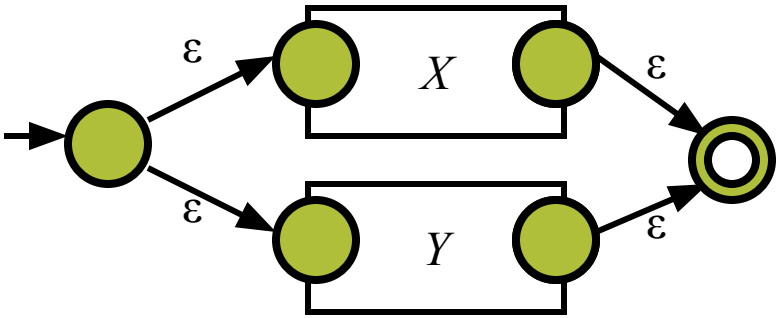
RE: *XY*
FA:



RE to NFA Conversion (Continue…)

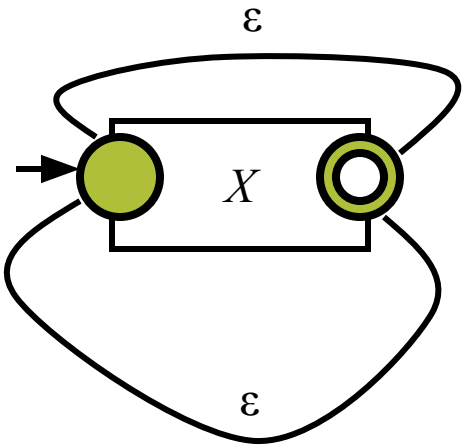
RE: $X|Y$

FA:



RE: X^*

FA:



RE to NFA Conversion - Example

- Convert the following RE to NFA
  - a + b
  - ab
  - a^*
  - $a^* + ab$

The Parsing Process

The Parsing Process

- The *Parser* determines the structure of the program from the tokens produced by the Scanner
- Parser can be viewed as a function that takes as its input the sequence of tokens and produces as its output the syntax tree
- One problem that is more difficult for the parser than the scanner is the treatment of errors
  - In scanner, if a character cannot be a legal token then it is simple to generate an error token & consumes character
  - The parser not only needs to report an error but also it needs to recover from error and continue parsing

Context-Free Grammars (CFGs)

- A CFG is a specification for the syntactic structure of a programming language
- This specification is very similar to the specification of lexical structure of a language using regular expressions except that a CFG involves recursive rules

CFG *exp* → *exp op exp* | **(** *exp* **)** | ***number***

op → **+** | **-** | *****

RE ***number* = digit digit***

digit = 0|1|2|3|4|5|6|7|8|9

Specification of CFG Rules

- Like regular expressions, grammar rules are defined over an alphabet or set of symbols
  - In RE these symbols are usually **characters**
  - In CFG these symbols are usually **tokens**
- Given an alphabet, a context-free grammar rule in BNF consists of a string of symbols
  - The first name is a name of a structure
  - The second symbol is a meta-symbol "→"
  - The → symbol is followed by string of symbols each of which is a symbol from the alphabet, a name for a structure, or the meta-symbol "|"

Examples

- $E \rightarrow (E) \mid a$
- $E \rightarrow (E)$
- $A \rightarrow Aa \mid a$ (~a* having a>=1)
- $A \rightarrow aA \mid a$ (~a* having a>=1)
- $A \rightarrow aA \mid a \mid \varepsilon$

Context Free Grammars

- A context-free grammar, CFG is a collection of three things
  - An alphabet ∑ of letters called terminals from which strings of language are generated
  - A set of symbols called nonterminals, one of which is a symbol S is termed as the start symbol
  - A finite set of productions of the form
    - One nonterminal->finite string of terminals and/or Nonterminals

Context Free Grammars (Continue…)

- The strings of terminals and nonterminals can consist of only terminals or of only nonterminals, or of any mixture of terminals and nonterminals or even the empty string
- A CFG must has at least one production that has the nonterminal S at its left side

Context Free Grammars (Continue…)

- Non-terminal/Variables/Syntactic category
  - A symbol that will be substituted by some other symbol(s)
  - Variable because the same non-terminal can have multiple substitutions
- Terminal
  - A symbol that cannot be substituted further
  - Letter from the alphabet set

Context Free Grammars (Continue…)

- Consists of language sub-elements (Syntactic categories)
  - Each syntactic category defines a subset of the language
- Relating these syntactic category gives the definition of the language itself
- Start Symbol (non-terminal) gives the entry point for word generation

Context Free Grammars (Continue…)

- Conventions for CFG
  - Nonterminals are written as capital letters
  - Terminals Symbols are written in lower case
- Terminal symbols are also called atomic symbols

Context Free Grammars (Continue…)

- Terminologies
  - Generation or Derivation
    - The sequence of applications of the rules that produces the finished string of terminals from the starting symbol is called a derivation or a generation of the word
  - Production
    - The grammatical rules are called productions

Context Free Languages

- The language generated by a CFG is the set of all strings of terminals that can be produced from the start symbol S using the productions as substitutions.
- A language generated by a CFG is called a context free language (CFL)

CFG - Examples

- Examples
  - All strings having even a's and odd b's
  - All strings that start and end with different letters
  - $a^n b^n$
  - $a^n b^m a^n$

Derivations

exp → *exp op exp* | **(** *exp* **)** | ***number***
op → **+** | **-** | *****

Derivation of ["`(34-3)*42`"]

(1) *exp* ⇒ *exp op exp* [*exp* → *exp op exp*]
(2) ⇒ *exp op* `number` [*exp* → `number`]
(3) ⇒ *exp* `*` `number` [*op* → `*`]
(4) ⇒ `(` *exp* `)` `*` `number` [*exp* → `(` *exp* `)`]
(5) ⇒ `(` *exp op exp* `)` `*` `number` [*exp* → *exp op exp*]
(6) ⇒ `(`*exp op* `number``)` `*` `number` [*exp* → `number`]
(7) ⇒ `(`*exp* `-` `number``)` `*` `number` [*op* → `-`]
(8) ⇒ `(``number` `-` `number``)`*`number` [*exp* → `number`]

Derivations - Example

exp → *exp op exp* | **(** *exp* **)** | ***number***
op → **+** | **-** | *****

- Write down derivation of
  - **(((34-3)))**
  - **((44 + 4) -3)**

Summary

Any Questions?