CMSC 330: Organization of Programming Languages

Context-Free Grammars

Motivation

- Programs are just strings of text
  - But they're strings that have a certain structure
- Informal description of syntax of a C program
  - A C program is a list of declarations and definitions
  - A function definition contains parameters and a body
  - A function body is a sequence of statements
  - A statement is an expression, if, goto, etc.
  - An expression may be assignment, addition, subtraction, etc

Motivation (cont'd)

- We want to describe program structure precisely
- Regular expressions are not enough
  - No regular expression for balanced pairs of ( )'s
    - {"()", "(())", "()()", ... } is not a regular language
- Instead, we'll use context-free grammars
  - These are *almost* enough for C, C++, Java

Context Free Grammar (CFG)

- A way of generating sets of strings or languages
- Grammar: $S \rightarrow 0S \mid 1S \mid \varepsilon$
  - Means every S may be replaced by 0S, 1S, or ε
  - Example
    - $S \Rightarrow 0S$ // using $S \rightarrow 0S$
      $\Rightarrow 01S$ // using $S \rightarrow 1S$
      $\Rightarrow 011S$ // using $S \rightarrow 1S$
      $\Rightarrow 011$ // using $S \rightarrow \varepsilon$
- Grammar is same as regular expression (0|1)*
  - Generates / accepts the same set of strings

Context-Free Grammars (CFGs)

- But CFGs can do a lot more!
  - S → (S) | ε // generates balanced pairs of ( )'s
- In fact, CFGs subsume REs, DFAs, NFAs
  - There is a CFG that generates any regular language
  - But REs are a better notation for regular languages
- CFGs can specify programming language syntax
  - CFGs (mostly) describe the parsing process

Formal Definition

- A context-free grammar G is a 4-tuple:
  - Σ – a finite set of *terminal* or *alphabet* symbols
    - Often written in lowercase
  - N – a finite, nonempty set of *nonterminal* symbols
    - Often written in uppercase
    - It must be that N ∩ Σ =
  - P – a set of *productions* of the form N → (Σ|N)*
    - Informally this means that the nonterminal can be replaced by the string of zero or more terminals or nonterminals to the right of the →
    - Can think of productions as rewriting rules
  - S N – the *start symbol*

Backus-Naur Form

- Context-free grammar production rules are also called Backus-Naur Form or BNF
  - A production like A → B c D is written in BNF as
    <A> ::= <B> c <D> (Non-terminals written with angle brackets and ::= instead of →)
  - Often used to describe language syntax
- BNF was designed by
  - John Backus
    - Chair of the Algol committee in the early 1960s
  - Peter Naur
    - Secretary of the committee, who used this notation to describe Algol in 1962

Informal Definition of Acceptance

- A string is accepted by a CFG if there is
  - Some sequence of applying productions (rewrites) starting at the start symbol that generates the string
- Example
  - Grammar: S → 0S | 1S | ε
  - Sequence generating the string 010
    - S ⇒ 0S ⇒ 01S ⇒ 010S ⇒ 010
- Terminology
  - Such a sequence of rewrites is a derivation or parse
  - Discovering the derivation is called parsing

Derivations

- Notation
  - $\Rightarrow$ indicates a derivation of one step
  - $\Rightarrow^+$ indicates a derivation of one or more steps
  - $\Rightarrow^*$ indicates a derivation of zero or more steps
- Example
  - $S \rightarrow 0S \mid 1S \mid \varepsilon$
- For the string 010
  - $S \Rightarrow 0S \Rightarrow 01S \Rightarrow 010S \Rightarrow 010$
  - $S \Rightarrow^+ 010$
  - $S \Rightarrow^* S$

Practice

- Try to make a grammar which accepts
  - $0^*|1^*$
  - 0^n1^n where $n \geq 0$
  - 0^n1^m where $m \leq n$

$S \rightarrow A \mid B$
$A \rightarrow 0A \mid \varepsilon$
$B \rightarrow 1B \mid \varepsilon$

$S \rightarrow 0S1 \mid \varepsilon$

$S \rightarrow 0S1 \mid 0S\ \varepsilon$

- Give some example strings from this language
  - $S \rightarrow 0 \mid 1S$
    - 0, 10, 110, 1110, 11110, …
  - What language is it?
    - 1^*0

Example

$S \rightarrow aS \mid T$
$T \rightarrow bT \mid U$
$U \rightarrow cU \mid \varepsilon$

- A derivation:
  - $S \Rightarrow aS \Rightarrow aT \Rightarrow aU \Rightarrow acU \Rightarrow ac$
    - Abbreviated as $S \Rightarrow^+ ac$
  - $S \Rightarrow T \Rightarrow U \Rightarrow \varepsilon$
- Is there any derivation
  - $S \Rightarrow^+ ccc$? $S \Rightarrow^+ Sa$?
  - $S \Rightarrow^+ bab$? $S \Rightarrow^+ bU$?

Example (cont'd)

$S \rightarrow aS \mid T$
$T \rightarrow bT \mid U$
$U \rightarrow cU \mid \varepsilon$

- Generates what language?
- Do other grammars generate this language?

$S \rightarrow ABC$
$A \rightarrow aA \mid \varepsilon$
$B \rightarrow bB \mid \varepsilon$
$C \rightarrow cC \mid \varepsilon$

  - So grammars are not unique