CMSC 330: Organization of Programming Languages

Context-Free Grammars

Reminders / Announcements

- Project 2 was posted on Sep. 24
- Class participation is part of your grade

Motivation

- Programs are just strings of text
  - But they're strings that have a certain structure
    - 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 either an expression, an if, a goto, etc.
    - An expression may be assignment, addition, subtraction, etc
- We want to solve two problems
  - We want to describe programming languages precisely
  - We need to describe more than the regular languages
    - Recall that regular expressions , DFAs, and NFAs are limited in their expressiveness

Program structure

Syntax

- What a program looks like
- BNF (context free grammars) - a useful notation for describing syntax.

Semantics

- Execution behavior

Context-Free Grammars (CFGs)

- A way of generating sets of strings or languages
- They subsume regular expressions (and DFAs and NFAs)
  - There is a CFG that generates any regular language
  - (But regular expressions are a better notation for languages which are regular.)
- They can be used to describe programming languages
  - They (mostly) describe the parsing process

Simple Example

$S \rightarrow 0|1|0S|1S|\varepsilon$

- This is the same as the regular expression (0|1)*
- But CFGs can do a lot more!

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 $N \cap \Sigma = \emptyset$
  - P – a set of *productions* of the form $N \rightarrow (\Sigma|N)^*$
    - Informally this means that the nonterminal can be replaced by the string of zero or more terminals or nonterminals to the right of the →
  - $S \in N$ – the *start symbol*

Informal Definition of Acceptance

- A string is accepted by a CFG if there is some path that can be followed starting at the start state which generates the string

Example:

S → 0|1|0S|1S|ε

0101:

S →0S →01S →010S →0101

Example: Arithmetic Expressions (Limited)

- E → a | b | c | E+E | E-E | E*E | (E)
  - An expression E is either a letter a, b, or c
  - Or an E followed by + followed by an E
  - etc.
- This describes or generates a set of strings
  - {a, b, c, a+b, a+a, a*c, a-(b*a), c*(b + a), …}
- Example strings not in the language
  - d, c(a), a+, b**c, etc.

Formal Description of Example

- Formally, the grammar we just showed is
  - Σ = { +, -, *, (,), a, b, c }
  - N = { E }
  - P = { E → a, E → b, E → c, E → E-E, E → E+E, E → E*E, E → (E)}
  - S = E

Notational Shortcuts

- If not specified, assume the left-hand side of the first listed production is the start symbol
- Usually productions with the same left-hand sides are combined with |
- If a production has an empty right-hand side it means ε

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
- 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

Uniqueness of Grammars

- Grammars are not unique. Different grammars can generate the same set of strings.
- The following grammar generates the same set of strings as the previous grammar:
  - $E \rightarrow E+T \mid E-T \mid T$
  - $T \rightarrow T*P \mid P$
  - $P \rightarrow (E) \mid a \mid b \mid c$

Another Example Grammar

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

What are some strings in the language?

Practice

Try to make a grammar which accepts…

- 0*|1*
- $a^n b^n$

Give some example strings from this language:

- $S \rightarrow 0 \mid 1S$

What language is it?

Sentential Forms

A *sentential form* is a string of terminals and nonterminals produced from the start symbol

Inductively:

- The start symbol
- If $\alpha A\delta$ is a sentential form for a grammar, where (α and $\delta \in (N|\Sigma)^*$), and $A \rightarrow \gamma$ is a production, then $\alpha\gamma\delta$ is a sentential form for the grammar
  - In this case, we say that $\alpha A\delta$ *derives* $\alpha\gamma\delta$ in one step, which is written as $\alpha A\delta \Rightarrow \alpha\gamma\delta$

Derivations

- $\Rightarrow$ is used to indicate a derivation of one step
- $\Rightarrow^+$ is used to indicate a derivation of one or more steps
- $\Rightarrow^*$ indicates a derivation of zero or more steps

Example:
$S \rightarrow 0|1|0S|1S|\varepsilon$

0101:
$S \Rightarrow 0S \Rightarrow 01S \Rightarrow 010S \Rightarrow 0101$
$S \Rightarrow^+ 0101$
$S \Rightarrow^* S$

Language Generated by Grammar

A slightly more formal definition…

- The language defined by a CFG is the set of all sentential forms made up of only terminals.

Example:
$S \rightarrow 0|1|0S|1S|\varepsilon$

In language:	Not in language:
01, 000, 11, ε …	0S, a, 11S, …

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 $S \Rightarrow^+ ac$
    - So S, aS, aT, aU, acU, ac are all sentential forms for this grammar
  - $S \Rightarrow T \Rightarrow U \Rightarrow \varepsilon$
- Is there any derivation
  - $S \Rightarrow^+ ccc$? $S \Rightarrow^+ Sa$?
  - $S \Rightarrow^+ bab$? $S \Rightarrow^+ bU$?

The Language Generated by a CFG

- The *language generated by a grammar G* is

$$L(G) = \{ \omega \mid \omega \in \Sigma^* \text{ and } S \Rightarrow^+ \omega \}$$

  - (where S is the start symbol of the grammar and Σ is the alphabet for that grammar)
- I.e., all sentential forms with only terminals
- I.e., all strings over Σ that can be derived from the start symbol via one or more productions

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

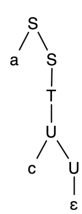
Parse Trees

- A *parse tree* shows how a string is produced by a grammar
  - The root node is the start symbol
  - Each interior node is a nonterminal
  - Children of node are symbols on r.h.s of production applied to that nonterminal
  - Leaves are all terminal symbols
- Reading the leaves left-to-right shows the string corresponding to the tree

Example

$S \Rightarrow aS \Rightarrow aT \Rightarrow aU \Rightarrow acU \Rightarrow ac$

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

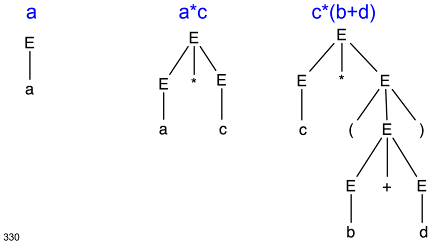


Parse Trees for Expressions

- A *parse tree* shows the structure of an expression as it corresponds to a grammar

$E \rightarrow a \mid b \mid c \mid d \mid E+E \mid E-E \mid E*E \mid (E)$

a a*c c*(b+d)



Practice

E → a | b | c | d | E+E | E-E | E*E | (E)

Make a parse tree for…

- a*b
- a+(b-c)
- d*(d+b)-a
- (a+b)*(c-d)
- a+(b-c)*d