CS280 Study Guide : Exam 1 : Chapters 3-5
Scheduled: June 19th

Chapter 3

syntax - the form of structure of the expressions, statements, and program units
semantics - the meaning of the expressions, statements, and program units
lexeme - lowest level syntactic unit
recognizers - recognition device reads input strings over the alphabet of the language and decides whether the input strings belong to the language

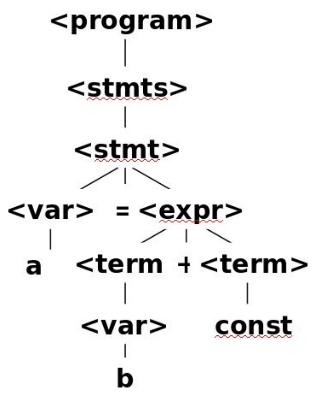
<u>Example BNF Grammar</u>

```
<program>  ->  <stmts>
<stmts>    ->  <
<stmt>     ->  <var> = <expr>
<var>      ->  a | b | c | d
<expr>     ->  <term> + <term> | <term> - <term>
<term>     ->  <var> | const
```

<u>Example Derivation</u>

```
<program>  => <stmts> => <stmt>
     |                  => <var> = <expr>
     |                  => a = <expr>
     |                  => a = <term> + <term>
     |                  => a = <var> + <term>
     V                  => a = b + <term>
Becomes "sentence"      => a = b + const
```

<u>Example Parse Tree</u>



A grammar is ambiguous if and only if it generates a sentential form(“sentence”) that has two or more distinct parse tree.

Difference between BNF and EBNF: (basically more complex w/ more customization)

- EBNF has optional parts placed in brackets [ ]
- EBNF has alternative parts placed in parentheses (+ | -)
- EBNF has repetitions (0 or more) placed in braces: <ident> -> letter {letter | digit}

Grammars are well-suited for describing the syntax of programming languages.
Attribute grammar is a descriptive grammar that can describe both the syntax and the semantics of a language.
Three primary methods of semantic description:

- Operation
- Axiomatic
- Denotational

operational - state changes are defined by coded algorithms
denotational - state changes are defined by rigorous mathematical functions

Chapter 4

If a grammar has left recursion, either direct or indirect, it cannot be the basis for a top down parser.
Useful parsers only look one token ahead.

Top-Down Parser
The parser must choose the correct rule in the leftmost derivation using only the first token produced.
Produces a parse tree that begins at the root and ends with the leaves.

Lexical analysis is the first phase of a compiler. Takes code in form of sentences. The lexical analyzer breaks these down to tokens (removing any whitespace or comments.)

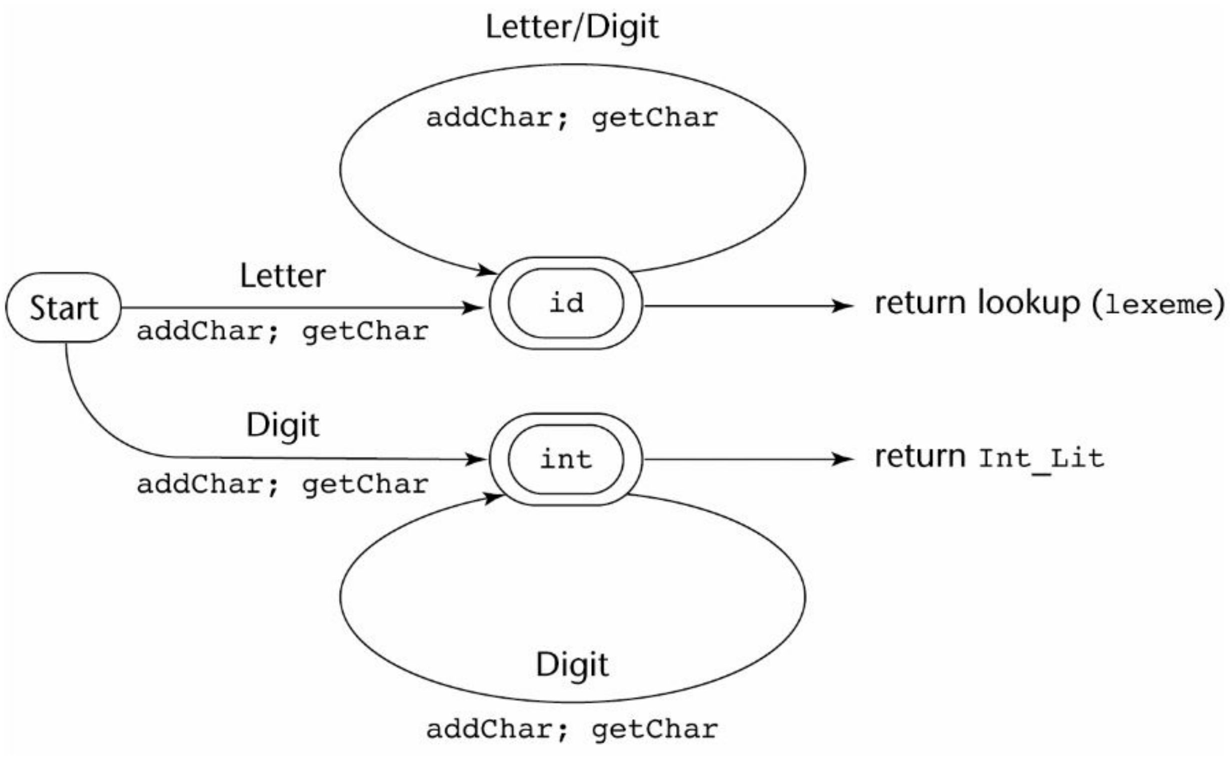
Syntax analysis, or parser, takes the input from a lexical analyzer in form of token streams. Checks for errors. Output is a parse tree.

A recursive-descent parser is an LL parser (EBNF.)

Parsing Problem (Bottom-up) - find the substring of current sentential form

LR family of shift-reduce parsers is the most common bottom-up parsing approach.

State Diagram (example)



Chapter 5

binding - is an association between an entity and an attribute, such as between an operation and a symbol

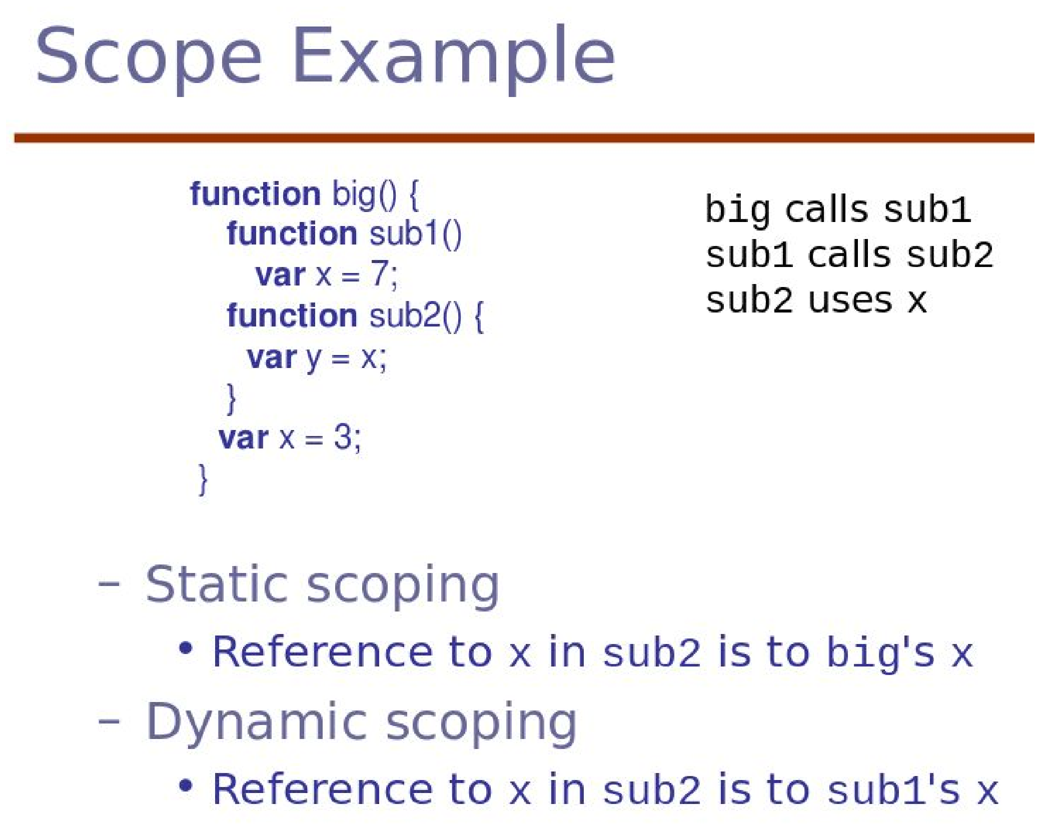
Binding is the association of attributes with program entities.

binding time - the time at which a binding takes place

static - if binding occurs before runtime and remains unchanged throughout program execution

dynamic - if binding occurs during execution or can change during execution of the program

scope - of a variable is the range of statements over which it is visible


Scope Example

```
function big() {
   function sub1()
     var x = 7;
   function sub2() {
     var y = x;
   }
  var x = 3;
}
```

```
big calls sub1
sub1 calls sub2
sub2 uses x
```

- Static scoping
  - Reference to x in sub2 is to big's x
- Dynamic scoping
  - Reference to x in sub2 is to sub1's x

Variables are characterized by:

- Name, address, value, type, lifetime, scope

Scalar Variables:

- Static, stack dynamic, explicit heap dynamic, implicit heap dynamic

Strong typing means detecting all type errors.

Compile time vs. Runtime

The source code must be compiled into machine code in order to become an executable program. The compilation process is referred to as compile time. A compiled program can be opened and run by a user. When an application is running, it is called runtime.