Today’s topics

Java
Syntax of Computer Language

Upcoming
More Java

Reading
Great Ideas, Chapter 2

Grammar

- English and other natural languages have structure
  \[ <S> \Rightarrow <\text{NOUN-PHRASE}> \text{ } <\text{VERB-PHRASE}> \]
  \[ <\text{NOUN-PHRASE}> \Rightarrow <\text{NOUN}> \text{ } | \text{ } <\text{ARTICLE}> <\text{NOUN}> \]
  \[ <\text{VERB-PHRASE}> \Rightarrow <\text{VERB}> \text{ } | \text{ } <\text{VERB}> <\text{NOUN-PHRASE}> \]
  \[ <\text{NOUN}> \Rightarrow \text{DOG} \text{ } | \text{ } \text{FLEAS} \text{ } | \text{ } \text{PERSON} \text{ } | \ldots \]
  \[ <\text{VERB}> \Rightarrow \text{RAN} \text{ } | \text{ } \text{BIT} \text{ } | \ldots \]

- Process of taking sentence and fitting it to grammar is called parsing

  \[ \text{DOG} \text{ } \text{BIT} \text{ } \text{PERSON} \]
  \[ <\text{NOUN}> <\text{VERB}> <\text{NOUN}> \text{ } <\text{NOUN}-\text{PHRASE}> <\text{VERB}-\text{PHRASE}> <\text{S}> \]

- Parsing English is complex because of context dependence

Formal specifications

- Need a precise notation of syntax of a language
- Grammars can also be used for generation
- Grammar rule:
  \[ <\text{name}> \Rightarrow <\text{sequence of letters and/or digits that begins with a letter}> \]
- Grammar use:
  \[ <\text{name}> \Rightarrow \text{guessB} \]
  \[ <\text{name}> \Rightarrow \text{msg42} \]
- Substitute as many times as necessary. All legal statements can be generated this way
- Want: \text{person} = \text{firstn} + " " + \text{lastn};
  - How do we get this from our grammar?

A Grammar for Java

- Need a set of rules
- Our first one was a good start:
  \[ <\text{name}> \Rightarrow \text{any string of alphanumeric symbols that begins with a letter} \]
- Let’s add something to define a simple statement:
  \[ <\text{statement}> \Rightarrow <\text{name}> = <\text{expression}> ; \]
- And then work on the details:
  \[ <\text{expression}> \Rightarrow <\text{string-expression}> | <\text{int-expression}> | <\text{others-expression}> \]
  \[ <\text{string-expression}> \Rightarrow <\text{string}> \]
  \[ <\text{string}> \Rightarrow <\text{name}> \]
  \[ <\text{string}> \Rightarrow " \text{any sequence of characters}" \]
A Simple Statement

- Now have enough to generate a statement like:
  ```
  msg = "hello";
  ```

- Start with:
  ```
  <statement> => <name> = <expression> ;
  ```
- Then using: `<name> = any string of alphanumeric symbols that begins with a letter`
  ```
  msg = <expression> ;
  ```
- Then, using: `<expression> = any string-expression | int-expression | oth-expression`
  ```
  msg = <string-expression> ;
  ```
- Using: `<string-expression> => <string>`
  ```
  msg = <string> ;
  ```
- Using: `<string> = any sequence of characters```
  ```
  msg = "hello" ;
  ```

A Grammar for Java

- Including more rules to describe programs we have:
  ```
  1. <name> => any string of alphanumeric symbols that begins with a letter
  2. <statement> => <name> = <expression> ;
  3. <statement> => <name> = new <class> (<arguments>) ;
  4. <statement> => <name> . <method> (<arguments>) ; | method (<arguments>) ;
  5. <arguments> => possibly empty list of <expression>s separated by commas
  6. <expression> => <string-expression> | <int-expression> | <oth-expression>
  7. <string-expression> => <string-expression> + <string-expression>
  8. <string-expression> => <string>
  9. <string> = any sequence of characters```
  ```
  10. <string> = <name>
  ```

Using our Grammar

- Use this to generate:
  ```
  person = firstn + " " + lastn;
  ```

<table>
<thead>
<tr>
<th>Rule</th>
<th>Statement being Generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>#2</td>
<td>&lt;statement&gt; =&gt; &lt;name&gt; = &lt;expression&gt; ;</td>
</tr>
<tr>
<td>1</td>
<td>&lt;statement&gt; =&gt; &lt;name&gt; = &lt;expression&gt; ;</td>
</tr>
<tr>
<td>6</td>
<td>&lt;statement&gt; =&gt; &lt;name&gt; = &lt;string-expression&gt; ;</td>
</tr>
<tr>
<td>7</td>
<td>&lt;statement&gt; =&gt; &lt;name&gt; = &lt;string-expression&gt; + &lt;string-expression&gt; ;</td>
</tr>
<tr>
<td>8</td>
<td>&lt;statement&gt; =&gt; &lt;name&gt; = &lt;string&gt; + &lt;string-expression&gt; ;</td>
</tr>
<tr>
<td>10</td>
<td>&lt;statement&gt; =&gt; &lt;name&gt; = &lt;string&gt; + &lt;name&gt; ;</td>
</tr>
<tr>
<td>1</td>
<td>&lt;statement&gt; =&gt; &lt;name&gt; = &lt;string&gt; + &lt;string&gt; ;</td>
</tr>
</tbody>
</table>

Proving Grammatical Correctness

- Why go through the process we went through?
  - Shows that desired statement can be generated from this grammar
  - Actually proves that the statement is grammatically correct!
  - Same rigor as a mathematical proof
  - (Does not prove that logic is correct, though)

- Actually need more rules to handle the level of Java we’ve covered so far
  - Summary of rules shown on pages 78-80 of *Great Ideas*
  - Also gives an example for a complete applet
  - Too long to go through in class – Please Read!