Importance of Language

- Vehicle for programming
- Use for human and machine communications
- Syntax Rules
  - First pass already covered
  - Need to refine to notation
    - Must be suitable for machine to do
- In addition, need to deal with the meaning
- Also, should see Levels or Layers in dealing with computer
  1. Hardware
  2. Machine language
  3. Assembler
  4. Java (or other high-level language)
  5. Application (e.g. Word, Excel, Filemaker, …)
Language Translation

- **Goal** is to automatically
  - translate Java:
    
    \[ z = x + y; \]
  - to Assembler:
    
    ```
    copy ax, x
    add ax, y
    copy z, ax
    ```

- What is the *meaning* we are looking for?
  
  Machine gives assembler statements meaning because the machine knows what to do with them (after trivial translation to binary). E.g., the machine knows what **add** means.
Revise Syntactic Rules

- Need to revise Syntactic Production Rules
  - New rule:
    R1: \(<n>j \rightarrow a\ sequence\ of\ letters\ and/or\ digits\ that\ been\ with\ a\ letter\)
  - Replaces (have seen these before)
    R1: \(<name> \rightarrow a\ sequence\ of\ letters\ and/or\ digits\ that\ been\ with\ a\ letter\)
  - The new R1 says “change \(<n>j\) into a sequence of letters and/or digits that begin with a letter”

- Use rules to modify strings
  - For syntactic productions, must end up with valid Java Programs
Using Syntax Rules

- Examples using R1:
  \[<n>3 \rightarrow x]\n  Or
  \[<n>6 \rightarrow \text{data}\]
  Where “n” stand for “name”

- Further use of R1:
  \[(<n>3 + <n>6)\]
  Use \(<n>3\) and \(<n>6\) above to get
  \[(x + \text{data})\]

- More Rules:
  - R2 : \(<e> \rightarrow <n>j\)
    Where “e” stands for “expression”
  - Example:
    \(<e>1 \rightarrow <n>3\)
Using Syntax Rules

- and
  - R3: <s>k -> <n>j = <e>i ;
  - Where “s” stands for “statement”
  - It says “<s>k” can be replaced by “<n>j = <e>i ;”
- Can now do: ans = data;

  derivation rule

  <s>1      R3: <s>1 -> <n>2 = <e>3 ;
  <n>2 = <e>3 ;       R1: <n>2 -> ans
  ans = <e>3 ;       R2: <e>3 -> <n>4
  ans = <n>4 ;      R1: <n>4 -> data
  ans = data ;
More Rules

- Need two more rules to make it worthwhile
  - R4: $<e>i \rightarrow ( <e>j + <e>k )$
  - R5: $<e>i \rightarrow ( <e>j * <e>k )$

These are additional rules for expressions

- Can now handle $\text{ANS} = (X + (Y * Z))$;
  (notice shorthand/simplification used)
## Longer Example

\[
\text{ANS} = (X + (Y * Z)) ;
\]

<table>
<thead>
<tr>
<th>derivation</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>R3:  s1 -&gt; n2 = e3;</td>
</tr>
<tr>
<td>n2 = e3;</td>
<td>R1:  n2 -&gt; ANS</td>
</tr>
<tr>
<td>ANS = e3;</td>
<td>R4:  e3 -&gt; (e4 + e5)</td>
</tr>
<tr>
<td>ANS = (e4 + e5);</td>
<td>R2:  e4 -&gt; n6</td>
</tr>
<tr>
<td>ANS = (n6 + e5);</td>
<td>R1:  n6 -&gt; X</td>
</tr>
<tr>
<td>ANS = (X + e5);</td>
<td>R5:  e5 -&gt; (e7 * e8)</td>
</tr>
<tr>
<td>ANS = (X + (e7 * e8));</td>
<td>R2:  e7 -&gt; n9</td>
</tr>
<tr>
<td>ANS = (X + (n9 * e8));</td>
<td>R1:  n9 -&gt; Y</td>
</tr>
<tr>
<td>ANS = (X + (Y * e8));</td>
<td>R2:  e8 -&gt; n10</td>
</tr>
<tr>
<td>ANS = (X + (Y * n10));</td>
<td>R1:  n10 -&gt; Z</td>
</tr>
<tr>
<td>ANS = (X + (Y * Z));</td>
<td></td>
</tr>
</tbody>
</table>

CPS 001
Notes

- Abbreviations
  - Just omitted the angle brackets. Could do this because the notation remained unambiguous.

- Role of the subscripts
  - The subscripts are required to make sure each term is unique.
  - Simplest technique is to simply start at one and increment every time another subscript is specified.

- Simple substitution is all that is required
  - If you are doing something more than that, it is probably wrong!

- The notation and form are important
  - You will be expected to match them on tests.