Today’s topics

Language Translation
  Revising the Syntax Rules and Notation
  Generating Code

Upcoming
  Computer Security

Reading
  Great Ideas, Chapters 11
Importance of Language

- Vehicle for programming
- Use for human and machine communications

Syntax Rules

- First pass already covered
- Need to refine the notation
  - Must be suitable for machine to use
- 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 from **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: \(<n>j\) -> a sequence of letters and/or digits that begin with a letter
  - Replaces (have seen this before):
    R1: \(<name>\) -> a sequence of letters and/or digits that begin 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:**
  
  $<n>_3 \rightarrow x$
  
  or
  
  $<n>_6 \rightarrow \text{data}$

  where "n" stand for "name"

- **Further use of R1:**
  
  $(<n>_3 + <n>_6)$
  
  Replace $<n>_3$ above to get
  
  $(x + <n>_6)$
  
  and $<n>_6$ to get
  
  $(x + \text{data})$

- **More Rules:**
  
  - R2: $<e>_i \rightarrow <n>_j$
    
    Where "e" stands for "expression"
  
  - Example:
    
    $<e>_1 \rightarrow <n>_3$
Using Syntax Rules

- and

  - R3: \(<s>_k \rightarrow <n>_j = <e>_i ;\)
  - Where “s” stands for “statement”
  - It says “\(<s>_k\)” can be replaced by “\(<n>_j = <e>_i ;\)”

- Can now do:  \(\text{ans} = \text{data};\)

**derivation**  |  **rule**
---|---
\(<s>_1\)  |  R3: \(<s>_1 \rightarrow <n>_2 = <e>_3;\)
\(<n>_2 = <e>_3;\)  |  R1: \(<n>_2 \rightarrow \text{ans}\)
\(\text{ans} = <e>_3;\)  |  R2: \(<e>_3 \rightarrow <n>_4\)
\(\text{ans} = <n>_4;\)  |  R1: \(<n>_4 \rightarrow \text{data}\)
\(\text{ans} = \text{data};\)
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/simplifications used)
Longer Example

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

derivation

\[
\begin{align*}
\text{s1} \\
n2 &= e3 ; \\
\text{ANS} &= e3 ; \\
\text{ANS} &= (e4 + e5) ; \\
\text{ANS} &= (n6 + e5) ; \\
\text{ANS} &= (X + e5) ; \\
\text{ANS} &= (X + (e7 \times e8)) ; \\
\text{ANS} &= (X + (n9 \times e8)) ; \\
\text{ANS} &= (X + (Y \times e8)) ; \\
\text{ANS} &= (X + (Y \times n10)) ; \\
\text{ANS} &= (X + (Y \times Z)) ;
\end{align*}
\]

rule

\[
\begin{align*}
\text{R3: s1} &\rightarrow n2 = e3 ; \\
\text{R1: n2} &\rightarrow \text{ANS} \\
\text{R4: e3} &\rightarrow (e4 + e5) \\
\text{R2: e4} &\rightarrow n6 \\
\text{R1: n6} &\rightarrow X \\
\text{R5: e5} &\rightarrow (e7 \times e8) \\
\text{R2: e7} &\rightarrow n9 \\
\text{R1: n9} &\rightarrow Y \\
\text{R2: e8} &\rightarrow n10 \\
\text{R1: n10} &\rightarrow Z
\end{align*}
\]
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 start subscripts at one and increment every time a different subscript is needed.

❖ **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.
Adding Semantics

- **Need to add semantic components to our rules**
  - For every syntax rule, we will add a semantic rule
  - This will show the assembler code generated
  - The code, as interpreted by the machine will provide the meaning

- **Revise R1**

<table>
<thead>
<tr>
<th>Syntax Rule</th>
<th>Semantic Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1: (&lt;n&gt;)j (\rightarrow) w</td>
<td>M(&lt;n&gt;j) = w</td>
</tr>
</tbody>
</table>

- M ... Meaning of ... Name ... Memory location
- In other words, use same identifier/name in both Java and Assembler
Adding Semantics

❖ Revise R2

Syntax Rule

R2: \( <e>_i \rightarrow <n>_j \)

Semantic Rules

\[ M(<e>_i) = M(<n>_j) \]

\[ \text{code}(<e>_i) = \text{nothing} \]

☐ No code is generated!

❖ Revise R3

Syntax Rule

R3: \( <s>_k \rightarrow <n>_j = <e>_i ; \)

Semantic Rules

\[ \text{code}(<s>_k) = \text{code}(<e>_i) \]

\[ \text{COPY AX, M(<e>_i)} \]

\[ \text{COPY M(<n>_j), AX} \]

☐ Says code for statement is code to calculate expression \( <e>_i \) and code to copy it into memory associated with \( <n>_j \)
Generating Code for \( X = Y \);

- Now have enough to demonstrate simplest case
  - Use syntactic production to *control* process
  - Associated semantic rules are applied at each step
- Use rules to generate code for \( X = Y \);

<table>
<thead>
<tr>
<th>Derivation</th>
<th>Syntax Rule</th>
<th>Semantic Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>R3: ( s1 \rightarrow n2 = e3 ); ( \text{code}(s1) = \text{code}(e3) )</td>
<td>COPY AX, M(e3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COPY M(n2), AX</td>
</tr>
</tbody>
</table>

**MEANING:** \( \text{code}(s1) = \text{code}(e3) \)

<table>
<thead>
<tr>
<th>n2 = e3;</th>
<th>R1: ( n2 \rightarrow X )</th>
<th>M(n2) = X</th>
</tr>
</thead>
</table>

**MEANING:** \( \text{code}(s1) = \text{code}(e3) \)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>COPY AX, M(e3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>COPY X, AX</td>
</tr>
</tbody>
</table>

CompSci 001
## Generating Code for $X = Y$

<table>
<thead>
<tr>
<th>Derivation</th>
<th>Syntax Rule</th>
<th>Semantic Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X = e3;$</td>
<td>R2: $e3 \rightarrow n4$</td>
<td>$M(e3) = M(n4)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{code}(e3) = \text{nothing}$</td>
</tr>
</tbody>
</table>

**MEANING:** $\text{code}(s1) = \text{nothing}$

- $\text{COPY AX, } M(n4)$
- $\text{COPY X, AX}$

<table>
<thead>
<tr>
<th>Derivation</th>
<th>Syntax Rule</th>
<th>Semantic Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X = n4;$</td>
<td>R1: $n4 \rightarrow Y$</td>
<td>$M(n4) = Y$</td>
</tr>
</tbody>
</table>

**MEANING:** $\text{code}(s1) = \text{COPY AX, Y}$

- $\text{COPY X, AX}$
More Rules

- **Revise R4**

**Syntax Rule**

\[ R4: <e>_i \rightarrow (<e>_j + <e>_k) \]

**Semantic Rules**

\[ M(<e>_i) = \text{createname} \]

\[ \text{code}(<e>_i) = \text{code}(<e>_j) + \text{code}(<e>_k) \]

\[ \text{COPY AX, } M(<e>_j) \]

\[ \text{ADD AX, } M(<e>_k) \]

\[ \text{COPY } M(<e>_i), \ AX \]

- Says code for \(<e>_i\) is code to calculate expression \(<e>_j\) followed by code to calculate expression \(<e>_k\) and code to add them together and store that sum into memory associated with \(<e>_i\)
More Rules

- **Revise R5**

  Syntax Rule  
  **R5**: \( <e>_i \rightarrow ( <e>_j \ast <e>_k ) \)

  Semantic Rules  
  \( M(<e>_i) = \text{createname} \)

  \( \text{code}(<e>_i) = \text{code}(<e>_j) \times \text{code}(<e>_k) \)

  \( \text{COPY AX, } M(<e>_j) \)

  \( \text{MUL AX, } M(<e>_k) \)

  \( \text{COPY } M(<e>_i), \ AX \)

- Says code for \( <e>_i \) is code to calculate expression \( <e>_j \) followed by code to calculate expression \( <e>_k \) and code to multiply them together and store that sum into memory associated with \( <e>_i \)

- Basically, rules R4 and R5 are identical except that the + and ADD in one are replaced by the * and MUL in the other.
## Code for $Z = (X + Y)$

<table>
<thead>
<tr>
<th>Derivation</th>
<th>Syntax Rule</th>
<th>Semantic Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>R3: $s1 \rightarrow n2=e3$; $\text{code}(s1) = \text{code}(e3)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{COPY AX, } M(e3)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{COPY M(n2), AX}$</td>
</tr>
<tr>
<td></td>
<td>MEANING: $\text{code}(s1) = \text{code}(e3)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{COPY AX, M(e3)}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{COPY Z, AX}$</td>
</tr>
<tr>
<td>n2 = e3;</td>
<td>R1: $n2 \rightarrow Z$</td>
<td>$M(n2) = Z$</td>
</tr>
<tr>
<td></td>
<td>MEANING: $\text{code}(s1) = \text{code}(e3)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{COPY AX, M(e3)}$</td>
</tr>
</tbody>
</table>
### Code for \( Z = (X + Y) ; \)

<table>
<thead>
<tr>
<th>Derivation</th>
<th>Syntax Rule</th>
<th>Semantic Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Z = e3 ; )</td>
<td>R4: ( e3 \rightarrow (e4+e5) )</td>
<td>( M(e3) = \text{CN1} )</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{code}(e3) &= \text{code}(e4) \\
\text{code}(e5) &= \text{code}(e5) \\
\text{COPY} &\ AX, M(e4) \\
\text{ADD} &\ AX, M(e5) \\
\text{COPY} &\ M(e3), AX
\end{align*}
\]

**MEANING:** \( \text{code}(s1) = \)

\[
\begin{align*}
\text{code}(e4) &\text{ code}(e5) \\
\text{COPY} &\ AX, M(e4) \\
\text{ADD} &\ AX, M(e5) \\
\text{COPY} &\ CN1, AX \\
\text{COPY} &\ AX, CN1 \\
\text{COPY} &\ Z, AX
\end{align*}
\]
Code for $Z = (X + Y);$. 

$Z = (e_4 + e_5);$  \hspace{1cm} R2: e_4 \rightarrow n_6 \hspace{1cm} M(e_4) = M(n_6) \hspace{1cm} code(e_4) = \textit{nothing}$

\begin{verbatim}
MEANING: code(s1) = code(e_5)
COPY AX, M(n_6)
ADD AX, M(e_5)
COPY CN1, AX
COPY AX, CN1
COPY Z, AX
\end{verbatim}

$Z = (n_6 + e_5);$  \hspace{1cm} R1: n_6 \rightarrow X \hspace{1cm} M(n_6) = X$

\begin{verbatim}
MEANING: code(s1) = code(e_5)
COPY AX, X
ADD AX, M(e_5)
COPY CN1, AX
COPY AX, CN1
COPY Z, AX
\end{verbatim}
Code for $Z = (X + Y)$; \[.4\]

$Z = (X + e5)$;

R2: $e5 \rightarrow n7$

$M(e5) = M(n7)$

$code(e5) =$ nothing

**MEANING:**

$code(s1) =$ nothing

COPY AX, X
ADD AX, $M(n7)$
COPY CN1, AX
COPY AX, CN1
COPY Z, AX

$Z = (X + n7)$;

R1: $n7 \rightarrow Y$

$M(n7) = Y$

**MEANING:**

$code(s1) =$

COPY AX, X
ADD AX, Y
COPY CN1, AX
COPY AX, CN1
COPY Z, AX

$Z = (X + Y)$;
Towards a Real Program

- More complicated statement:
  \[ U_1 = (X + (Y \times Z)) \];
  - Done on pages 277-279 in text
  - (Note that book uses \(<i>j\) where we used \(<n>j\))

- Rules for Looping Sequence of statements

- Rules 6 and 7: A sequence of statements

  **Syntax Rule**
  **Semantic Rules**

  R6: \(<q>_i \rightarrow <s>_j \)

  code(<q>_i) = code(<s>_j)

  \(<q>_k \)

  code(<q>_k)

  R7: \(<q>_i \rightarrow <s>_j \)

  code(<q>_i) = code(<s>_j)

  - Says code for a sequence of statements is the code for the first
    statement followed by the code for the next statement, etc.
  - Notice the recursive nature of these statements.
More Complicated Statements

- **Rule 8: Compound Statement**
  
  Syntax Rule
  \[ R8: <c>i \rightarrow \{ \\}
  \]
  
  Semantic Rules
  \[ code(<c>i) = code(<q>j) \]

- **Rule 9: While Statement**
  
  Syntax Rule
  \[ R9: <s>i \rightarrow \text{while} (<n>j < <e>k) \]
  
  Semantic Rules
  \[ M(<s>i) = \text{create}\text{name} \]
  \[ M'(<s>i) = \text{create}\text{name} \]
  
  \[ code(<s>i) =
  \]
  
  \[ M(<s>i) \text{ code}(e_k) \]
  \[ \text{COPY AX, } M(<n>j) \]
  \[ \text{CMP AX, } M(e_k) \]
  \[ \text{JNB M'(<s>i)} \]
  \[ \text{JMP M(<s>i)} \]
  \[ M'(<s>i) \text{ NO-OP} \]
Final Thoughts

- **Clean Up Translation**
  - Some code generated can be removed
  - Modern compilers spend a lot of effort optimizing
- **Important: Everything done by simple substitution**
- **Everything “adds up”**
  - code( { <s>_1;<s>_2;<s>_3 } )
    - is
      - code(<s>_1)
      - code(<s>_2)
      - code(<s>_3)