Lecture topics

I. Translation schemes
II. Abstract syntax trees
III. Tree traversals
IV. Tree normalization

I. Translation schemes

SDD = syntax-directed definition
    = grammar + rules defining attributes

Example SDD:

```
S -> E_1 := E_2;
1 if E B
 2 if E B
```

Kinds of attributes:

<table>
<thead>
<tr>
<th>Synthesized</th>
<th>Inherited</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computed from</td>
<td></td>
</tr>
<tr>
<td>Children</td>
<td>Siblings or parent</td>
</tr>
<tr>
<td>Implemented in</td>
<td></td>
</tr>
<tr>
<td>Parser actions or tree traversal</td>
<td>Usually</td>
</tr>
<tr>
<td>tree traversal</td>
<td></td>
</tr>
</tbody>
</table>

S-attribute definition

= SDD with only synthesized attributes

Example implementation (in Jazz!):

```
AssignStmt assignStmt =
& expr COLONED expr SEMI
& expr := new AssignStmt(e, r); 3

See example-ast-visitor (Tact.rats, Main.java)
```

II. Abstract syntax trees

<table>
<thead>
<tr>
<th>Parsed tree</th>
<th>AST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td>concrete</td>
</tr>
<tr>
<td>Punctuation</td>
<td>E_1 := E_2;</td>
</tr>
<tr>
<td>Pass-thru production</td>
<td>E</td>
</tr>
<tr>
<td>Implmented implicit</td>
<td>allocate objects in memory</td>
</tr>
</tbody>
</table>

Example:

```
class Semantics
```
L-attributed definition
- SDD whose attributes are either synthesized
  or inherited, depend only on parent
  or left siblings

Implementation choices for attributes
with visitor
- field of AstNode
- return value from "visit" method
- output of "print"
- appended to list in field of visitor
See example - ast-guitar (Visitor.java, SyntaxTree.java)

Tree normalization

SDD for parser, create "raw AST"

\[
\begin{align*}
P & \rightarrow L \\
L & \rightarrow FT \\
T & \rightarrow FT_1 \mid E
\end{align*}
\]

\[
\begin{align*}
P_D & = \text{new Program} (L, D) \\
L_D & = L.T_D \\
T_D & = \begin{cases} 
F.D & \\
T_1.D & T_1.D = T.D
\end{cases}
\]

Building desired AST while visiting raw AST

\[
\text{Program}_D = \text{new Program} (L.D, D.D)
\]

\[
\text{FunDefListHead} = (\text{FunDef}_D, D.D)
\]

\[
\text{FunDef}_D = (\text{FunDef}_D, D.D)
\]

\[
\text{FunDef}_D = (\text{FunDef}_D, D.D)
\]

See example - ast-guitar (TreeNormalization.java)

Reminders
- hw5 due Fri 10/2
- pr2 due Fri 10/28