1 Abstract Syntax Trees
2 Syntax-Directed Definitions
3 Synthesized Attributes
4 Inherited Attributes
5 Examples
6 Assignment
A parse tree shows the *grammatical* structure of a statement.

It includes all of the grammar symbols (terminals and nonterminals) that were encountered during parsing.
Abstract Syntax Trees

- An abstract syntax tree (AST) shows the *logical* structure of the statement.
- Each node represents an action to be taken by the program or an object to be acted upon.
- The syntax tree may introduce operations that were not in the source code or the grammar.
  - Dereferencing operations.
  - Type-casting operations.
  - Jump statements.
Example (Syntax Trees and Parse Trees)

Consider the statement \( a = 2 \times b + c; \)
Our TreeBuilder program (Assignment 9) will “convert” the parse tree into the syntax tree.

The parse tree never really exists, except insofar as the parser follows its logical order.

The TreeBuilder will simply build the syntax tree from the information obtained by the parser.

Then the code generator (Assignment 10) will write the assembly code from the syntax tree.
Recursive descent parsers generally create a single AST for the entire program.

They build the tree from root to leaf.

Our parser will begin with the leaf nodes.

The AST will be built from the bottom up.
Our parser will generate a separate AST for each statement.
It will create a list of ASTs.
This will allow us to generate assembly code as the ASTs are created.
The trees will be connected both sequentially and through jump statements.
Abstract Syntax Trees

Syntax-Directed Definitions

Synthesized Attributes

Inherited Attributes

Examples

Assignment
A syntax-directed definition (SDD) is a context-free grammar with attributes added to the grammar symbols.

- These attributes are stored in the nodes of the syntax tree.
Example (Syntax-Directed Definitions)

- Let the grammar be
  \[ E \rightarrow E + E \mid \text{num} \]
- Then \( E \) derives its value from the \text{num} tokens in the expression.
- This is expressed formally by the rules
  \[ E.\text{val} = E_1.\text{val} + E_2.\text{val} \]
  \[ E.\text{val} = \text{num}.\text{lexval} \]
In a syntax-directed definition, each node has
- A set of synthesized attributes, and
- A set of inherited attributes.
1. Abstract Syntax Trees
2. Syntax-Directed Definitions
3. Synthesized Attributes
4. Inherited Attributes
5. Examples
6. Assignment
Definition

A synthesized attribute of a grammar symbol is a property that is determined by the attributes of the symbols below it in the parse tree.

- In other words, if $A \rightarrow \alpha$ is a production, then $A$'s synthesized attributes are determined by the attributes of the symbols in $\alpha$. 
Synthesized Attributes

- If the AST represents a numerical expression, then the value of the root node is determined by the values of the nodes below it in the tree.
- Thus, the value of the root node is a synthesized attribute.
Example (Synthesized Attributes)

- The terminals get their values directly from the lexical analyzer.
- For example, a `num` token’s value attribute would be the numerical value of the string of digits in the token.
Example (Synthesized Attributes)

```
expr.val

expr1.val  +  expr2.val

num.lexval

100   +   250
```
Example

Example (Synthesized Attributes)

expr.val

expr₁.val + expr₂.val

num.lexval

num.lexval

from lexer

100

250

from lexer
Example

Example (Synthesized Attributes)

```
expr.val
  /\  
expr1.val + expr2.val
   /\    
num.lexval  num.lexval
  /  
100  250
```

expr1.val: synthesized from lexer
expr2.val: synthesized from lexer
num.lexval: from lexer

Robb T. Koether (Hampden-Sydney College) Abstract Syntax Trees Synthetic and Inherited Mon, Mar 16, 2015 18 / 38
Example

Example (Synthesized Attributes)

expr.val

expr1.val

num.lexval

100

textsynthesized
textfrom lexer
textsynthesized
textfrom lexer

textexpr2.val

textnum.lexval

250

textsynthesized
textfrom lexer

textsynthesized
textfrom lexer

expr1.val

+ 

expr2.val

num.lexval

100

250

Robb T. Koether  (Hampden-Sydney College)Abstract Syntax TreesSynthetic and Inherited Mon, Mar 16, 2015 18 / 38
Outline

1. Abstract Syntax Trees
2. Syntax-Directed Definitions
3. Synthesized Attributes
4. Inherited Attributes
5. Examples
6. Assignment
Inherited Attributes

**Definition**

An *inherited attribute* is a property of a symbol (node) that is determined by its parent node and its siblings in the parse tree.

- In other words, if $\beta$ is a symbol on the right side of the production $A \rightarrow \alpha \beta \gamma$, then $\beta$’s inherited attributes are determined by the attributes of $A$ and the other symbols in $\alpha$ and $\gamma$. 
Consider the grammar for a declaration containing one or more identifiers.

\[
dcl \rightarrow type\ list  \\
list \rightarrow list\ ,\ \id\ |\ \id \\
type \rightarrow \textbf{int} \mid \textbf{float}
\]
Example

Example (Inherited Attributes)

- For example, the declaration might be
  
  ```
  float a, b, c;
  ```

- The attribute “float” first appears as the type of the `float` token.

- From there it is passed to the identifiers `a`, then `b`, then `c`. 
Example

Example (Inherited Attributes)

```
dcl
  type.type
    FLOAT
  list.type
    list.type
    ,
    id2.type
    float
    id1.type
```
Example

Example (Inherited Attributes)

```
dcl
type.type
FLOAT
list.type,
synthesized
list.type id2.type
id1.type float
from lexer
```
Example

Example (Inherited Attributes)

```
dcl
type.type
FLOAT
list.type
,
synthesized
inherited
list.type id2.type
id1.type
inherited
inherited
float
from lexer
```
Some Questions

Questions

• In an expression tree, is the type of the expression at the root inherited or is it synthesized?
• Is the type used in an arithmetic operation an inherited attribute or an synthesized attribute of the operator?
• In an assignment statement, is the type assigned by the operator an inherited attribute or a synthesized attribute of the operator?
Outline

1. Abstract Syntax Trees
2. Syntax-Directed Definitions
3. Synthesized Attributes
4. Inherited Attributes
5. Examples
6. Assignment
Let the grammar be

\[ E \rightarrow E + T \]
\[ E \rightarrow T \]
\[ T \rightarrow T \ast F \]
\[ T \rightarrow F \]
\[ F \rightarrow (E) \]
\[ F \rightarrow \text{num} \]
Example (Synthesized Attributes)

- The attribute at every node is the value of the nonterminal.
- In every case, it is synthesized.

\[
\begin{align*}
E\text{.val} & = E\text{.val} + T\text{.val} \\
E\text{.val} & = T\text{.val} \\
T\text{.val} & = T\text{.val} \times F\text{.val} \\
T\text{.val} & = F\text{.val} \\
F\text{.val} & = E\text{.val} \\
F\text{.val} & = \text{num. lexval}
\end{align*}
\]
Example (Inherited Attributes)

Let the grammar be

\[
E \rightarrow T \ E' \\
E' \rightarrow + \ T \ E' \\
E' \rightarrow \varepsilon \\
T \rightarrow F \ T' \\
T' \rightarrow * \ F \ T' \\
T' \rightarrow \varepsilon \\
F \rightarrow ( \ E ) \\
F \rightarrow \text{num}
\]
Example (Inherited Attributes)

- The attribute at the nodes $E$, $T$, and $F$ is the value of the nonterminal.
- In some cases, it is synthesized.
- In other cases, it is inherited.
Inherited Attributes

Example (Inherited Attributes)

- For the production
  
  \[ F \rightarrow \text{num} \]

  we have the rule

  \[ F.\text{val} = \text{num}.\text{lexval} \]

- For the production
  
  \[ F \rightarrow (\ E \ ) \]

  we have the rule

  \[ F.\text{val} = E.\text{val} \]

- After that, it is not so simple.
Example (Inherited Attributes)

Consider the parse tree for the expression

$$3 \times 4 + 5.$$
Inherited Attributes

Example (Inherited Attributes)

The parse tree
Inherited Attributes

Example (Inherited Attributes)

```
num 3
F
num 4
num
T' F
T E'
E +
F T'
E'
e
num 5
num
T' F
T e
E'
num gets its values from the lexer
```
Inherited Attributes

Example (Inherited Attributes)

\[
F \cdot \text{val} = \text{num} \cdot \text{lexval}
\]
Example (Inherited Attributes)

\[ T'.\text{inh} = F.\text{val} \]
Example (Inherited Attributes)

- How does $T$ (in the production $T \rightarrow F \ T'$) get its value?
- It must multiply 3 and 4 to get 12.
- So, first $T'$ inherits 3 from $F$.
- Then, in the production $T' \rightarrow \ast F \ T'_1$, $T'_1$ inherits 12 from $T'$ and $F$.
- Then, $T'$ turns around and synthesizes 12 from $T'_1$.
- Then, back in the production $T \rightarrow F \ T'$, $T$ synthesizes 12 from $T'$. 
Inherited Attributes

Example (Inherited Attributes)

\[ T' \text{.inh} = F \text{.val} \]
Inherited Attributes

Example (Inherited Attributes)

\[ T'_1 . \text{inh} = T' . \text{inh} \times F . \text{val} \]
Inherited Attributes

Example (Inherited Attributes)

\[ T' \text{.syn} = T' \text{.inh} \]
Inherited Attributes

Example (Inherited Attributes)

\[ T'.\text{syn} = T'_1.\text{syn} \]
Inherited Attributes

Example (Inherited Attributes)

\[ E \]
\[ E' \]
\[ T \]
\[ T' \]
\[ F \]
\[ F' \]
\[ * \]
\[ num \]
\[ 3 \]
\[ 4 \]
\[ 12 \]
\[ inh 3 \]
\[ syn 12 \]
\[ + \]
\[ T \]
\[ E' \]
\[ ε \]
\[ T'.val = T'.syn \]

Robb T. Koether  (Hampden-Sydney College) Abstract Syntax Trees Synthetic and Inherited Attributes Mon, Mar 16, 2015 34 / 38
E\'.inh = T.val
Example (Inherited Attributes)

- We now have the rules

<table>
<thead>
<tr>
<th>Production</th>
<th>Semantic Rules</th>
</tr>
</thead>
</table>
| $T \rightarrow F\ T'$ | $T'.inh = F.val$
| | $T.val = T'.syn$
| $T' \rightarrow *\ F\ T'_1$ | $T'_1.inh = T'.inh \times F.val$
| | $T'.syn = T'_1.syn$
| $T' \rightarrow \varepsilon$ | $T'.syn = T'.inh$
| $F \rightarrow \text{num}$ | $F.val = \text{num}.lexval$
Inherited Attributes

Example (Inherited Attributes)

As well as the rules

<table>
<thead>
<tr>
<th>Production</th>
<th>Semantic Rules</th>
</tr>
</thead>
</table>
| $E \rightarrow T\ E'$ | $E'.inh = T.val$  
                     | $E.val = E'.syn$  |
| $E' \rightarrow + T\ E'_1$ | $E'_1.inh = E'.inh + T.val$  
                     | $E'.syn = E'_1.syn$  |
| $E' \rightarrow \varepsilon$ | $E'.syn = E'.inh$  |
Homework

- p. 309: 1, 2, 3