Compilation

Syntax-Directed Translation

Technique used to build semantic information for large structures, based on its syntax.

In a compiler, Syntax-Directed Translation is used for

- Constructing Abstract Syntax Tree
- Type checking
- Intermediate code generation

Scopes and Symbol Table
Another Example of Syntax-Directed Translation

```
Decl  →  Type VarList
Type  →  ...
VarList →  id , VarList
VarList →  id

Decl  →  Type VarList  {VarList,type := Type.type}
Type  →  ...  {Type.type := id.id}
VarList →  id , VarList  {VarList1.type := VarList.type;
                      id.type := VarList.type}
VarList →  id  {id.type := VarList.type}
```

CSE 394 Compiler Design XV·11

Attributes

- **Synthesized Attribute**: Value of the attribute computed from the values of attributes of grammar symbols on RHS.

  Example: `val` in Expression grammar

- **Inherited Attribute**: Value of attribute computed from values of attributes of the LHS grammar symbol.

  Example: `type` of `VarList` in declaration grammar

The Essence of Syntax-Directed Translation

The semantics (i.e., meaning) of the various constructs in the language is viewed as attributes of the corresponding grammar symbols.

Example:

sequence of characters 495
→ grammar symbol `TOK_INT`
→ meaning ≡ integer 495
→ is an attribute of `TOK_INT (jjval, int.val)`.

Attributes are associated with **Terminal** as well as **Nonterminal** symbols.

An Example of Syntax-Directed Translation

```
E  →  E * E  {E.val := E1.val * E2.val}
E  →  E + E  {E.val := E1.val + E2.val}
E  →  id     {E.val := int.val}
```

CSE 394 Compiler Design XV·9
Syntax-Directed Definition

*Actions* associated with each production in a grammar.

For a production $A \rightarrow X Y$, actions may be of the form:

- $A\text{.attr} := f(X\text{.attr}', Y\text{.attr}'')$ for synthesized attributes
- $Y\text{.attr} := f(A\text{.attr}', X\text{.attr}'')$ for inherited attributes

If the function $f$ does not have side effects, syntax directed definitions is also called *attribute grammars*.