# Wed 10/19/2011: Syntax-directed translation (p1)

## 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:

| | |
|---|---|
| $S \to E_1 := E_2;$ | S.a = new AssignStmt(E1.a, E2.a) |
| \| if E $B_1$ else $B_2$ | S.a = new IfStmt(E.a, B1.a, B2.a) |
| \| if E B | S.a = new IfStmt(E.a, B.a) |

Kinds of attributes:

| | Synthesized | Inherited |
|---|---|---|
| Computed from | Children | Siblings or parent |
| Implemented in | Parser actions or tree traversal | Usually, tree traversal |

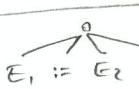
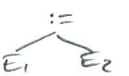
S-attributed definition
= SDD with only synthesized attributes

Example implementation (in Rats!):

```
· AssignStmt assignStmt =
    l:expr COLONEQ r:expr SEMI
    { yyValue = new AssignStmt(l, r); };
```

See example-ast-gen.tar (Tack.rats, Main.java)

## II Abstract syntax trees

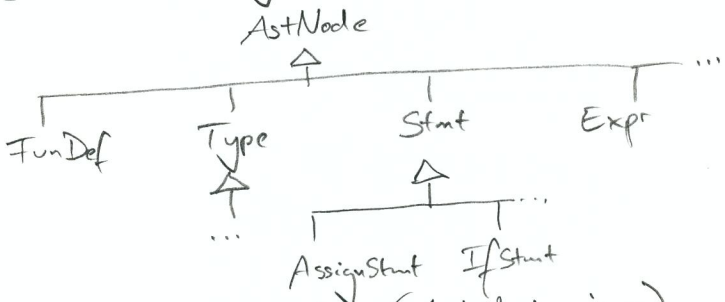
| | Parse tree | AST |
|---|---|---|
| Kind | concrete | abstract |
| Punctuation |   |   |
| Pass-thru production | E \| T \| F | F |
| Implemented | implicit | allocate objects in memory |

Example:

```
class AssignStmt extends Stmt {
    Expr _lhs;
    Expr _rhs;
    AssignStmt(Expr lhs, Expr rhs){
        _lhs = lhs; _rhs = rhs;
    }
    Object accept(Visitor v){
        return v.visit(this);
    }
}
```

(class, superclass, children; attributes would be in fields too)

Class hierarchy:




See example-ast-gen.tar (AstNode.java)

## III Tree traversals

Typically, left-to-right, depth-first

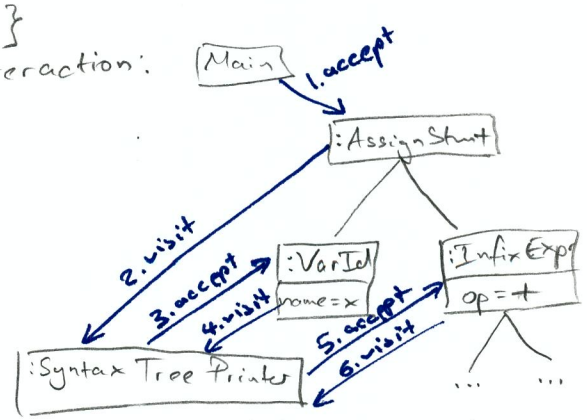
- tree normalizer
- tree printer
- scope analyzer
- type analyzer
- IR generator

} subclasses of Visitor

Example:

```
class SyntaxTreePrinter extends Visitor {
    Object visit(AssignStmt ast){
        print("AssignStmt");
        ast._lhs.accept(this);
        ast._rhs.accept(this);
    }
    ...
}
```

Interaction:




Double dispatch: call correct overloaded visit

Wed 10/19/2011: Syntax-directed translation (p.?)

L-attributed definition
= SDD where attributes are either synthesized or, if inherited, depend only on parent or left siblings

Implementation choices for attributes with visitors
- field of AstNode
- return value from "visit" method
- output of "print"
- appended to list in field of visitor

See example-ast-gen.tar (Visitor.java, SyntaxTreePrinter.java)

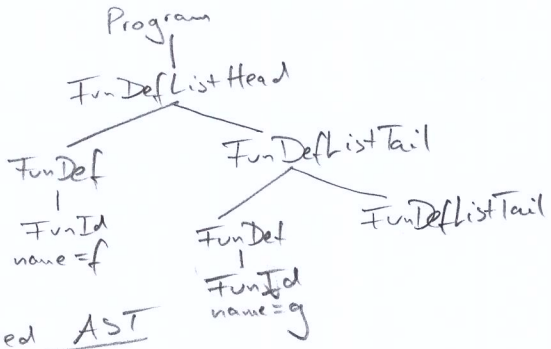
## IV Tree normalization

### SDD for parser, create "raw AST"

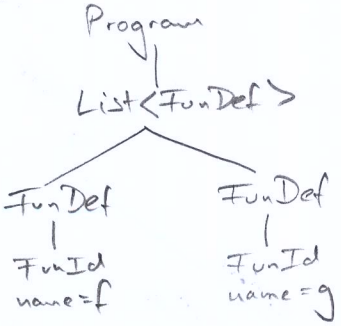
| | |
|---|---|
| $P \to L$ | $P.R = \text{new Program}(L.R)$ |
| $L \to F\ T$ | $L.R = \text{new FunDefListHead}(F.R, T.R)$ |
| $T \to F\ T_1$ | $T.R = \text{new FunDefListTail}(F.R, T_1.R)$ |
| $\mid \varepsilon$ | $T.R = \text{new FunDefListTail}()$ |

Input
f=fun
g=fun

Raw AST: (right-recursive)



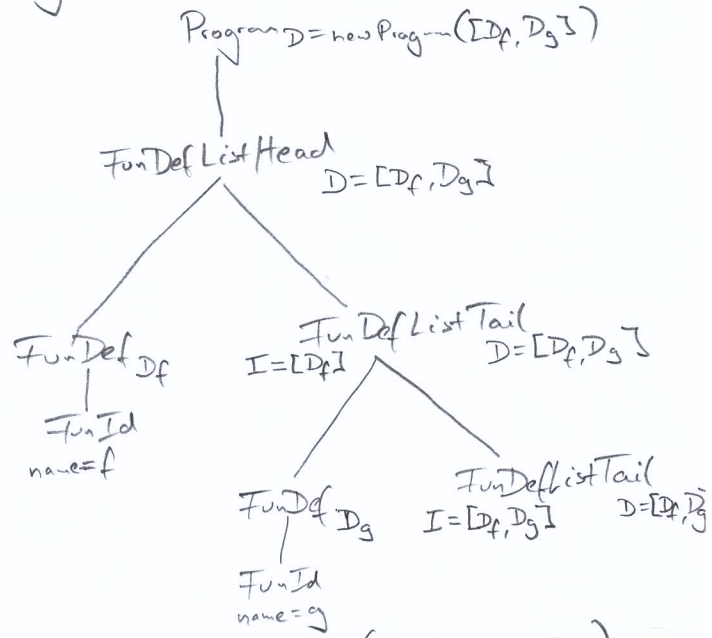

Desired AST




### SDD for normalizer visitor, creates Desired AST

| | |
|---|---|
| $P \to L$ | $P.D = \text{new Program}(L.D)$ |
| $L \to F\ T$ | $T.I = [F.D]$ ; $L.D = T.D$ |
| $T \to F\ T_1$ | $T_1.I = T.I \circ [F.D]$ ; $T.D = T_1.D$ |
| $\mid \varepsilon$ | $T.D = T.I$ |

### Building desired AST while visiting raw AST




See example-ast-gen.tar (TreeNormalizer.java) and Dragon Book Figure 5.13 (Page 321)

Reminders
- hw5 due Fr 10/21
- pr2 due Fr 10/28