

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:

$S \rightarrow E_1 := E_2$ | if $E B_1$ else B_2 | if $E B$
 $S.a = \text{new AssignStmt}(E.a, E.a)$ | $S.a = \text{new IfStmt}(E.a, B_1.a, B_2.a)$ | $S.a = \text{new IfStmt}(E.a, B.a)$

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!):

Assign Stmt assign Stmt =
 l: expr COLONEQ r: expr SEMI
 {yyval = new AssignStmt(l, r);};

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

II Abstract syntax trees

	Parse tree	AST
Kind	concrete	abstract
Punctuation	$E_1 := E_2 ;$	$E_1 := E_2$
Pass-thru production	$E \rightarrow T \mid F$	F
Implemented	implicit	allocate objects in memory

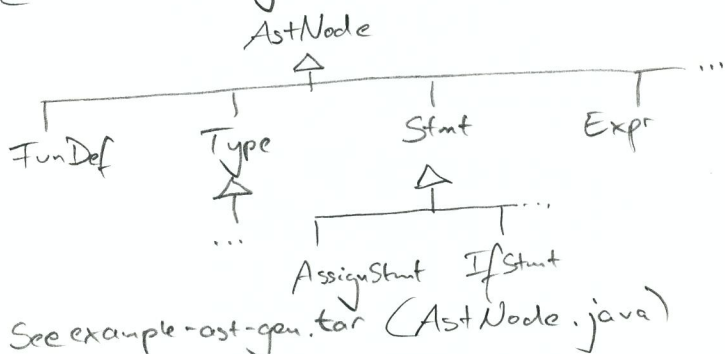
Example:

```

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);
    }
}
    
```

Annotations: class, superclass, children, attributes would be in fields too

Class hierarchy:



III Tree traversals

Typically, left-to-right, depth-first

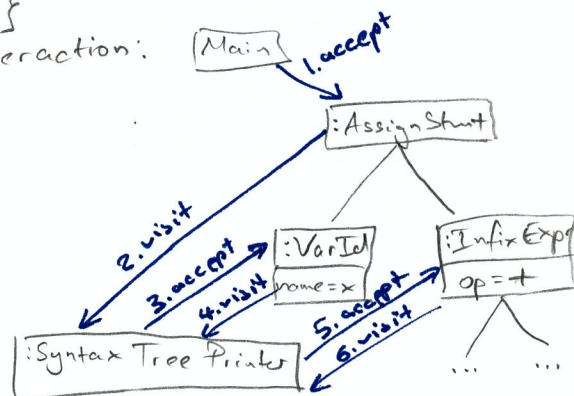
- tree normalizer
 - tree printer
 - scope analyzer
 - type analyzer
 - IR generator
- subclasses of Visitor

Example:

```

class SyntaxTreePrinter extends Visitor {
    Object visit(AssignStmt ast) {
        print("Assign Stmt");
        ast.-lhs.accept(this);
        ast.-rhs.accept(this);
    }
}
    
```

Interaction:



Double dispatch: call correct overloaded visit

Wed 10/13/2011: 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, SyntaxTreePrint.java)

IV Tree normalization

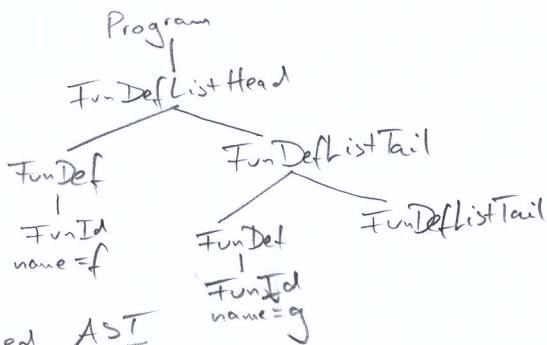
SDD for parser, create "raw AST"

$P \rightarrow L$	$P.R = \text{new Program}(L.R)$
$L \rightarrow FT$	$L.R = \text{new FunDefListHead}(F.R, T.R)$
$T \rightarrow FT_1$	$T.R = \text{new FunDefListTail}(F.R, T_1.R)$
$\mid \epsilon$	$T.R = \text{new FunDefListTail}()$

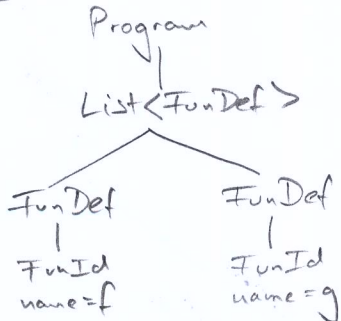
Input

f=fun
g=fun

Raw AST: (right-recursive)



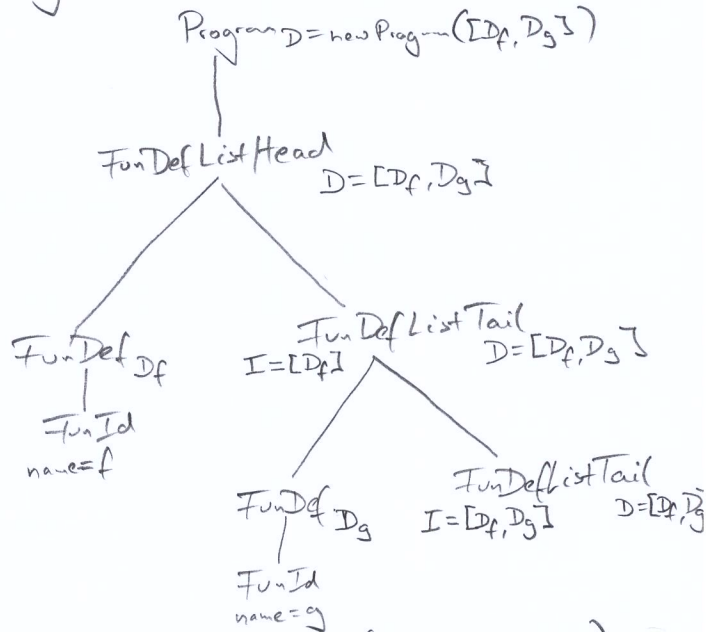
Desired AST



SDD for normalizer visitor, creates Desired AST

$P \rightarrow L$	$P.D = \text{new Program}(L.D)$
$L \rightarrow FT$	$T.I = [F.D] ; L.D = T.D$
$T \rightarrow FT_1$	$T.I = T.I \circ [F.D] ; T.D = T_1.D$
$\mid \epsilon$	$T.D = T.I$

Building desired AST while visiting raw AST



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

Reminders

- hw 5 due Fr 10/21
- pr 2 due Fr 10/28