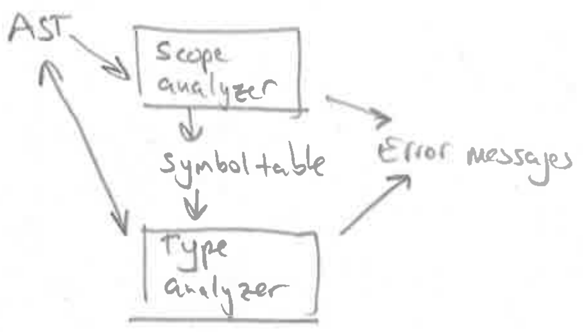
Name and type analysis pg. 1

Lecture topics:

I. Scopes and definitions
II. Scope analyzer
III. Types and their relations
IV. Type analyzer

Big picture



I Scopes and definitions

Definition	Example	Entity
FunDef	main = fun() → int { → 0; }	function main

VarDef	v = 1; #def
	v := v+1 ; # assign
		variable v

FieldLit	r = (f=1, g=2);
	print (r.f : string);
		fields f, g

FieldType	r = (f=1, g=2) : (f: int, g: int);	fields f, g
FieldType	f = fun (p:int, q:int) → int { → p+q ; }	Variables (parameters p, q)
For Stmt	for i in [1,2,3] { print (i: string); }	Variable (iterator i)

Lexical scoping example:

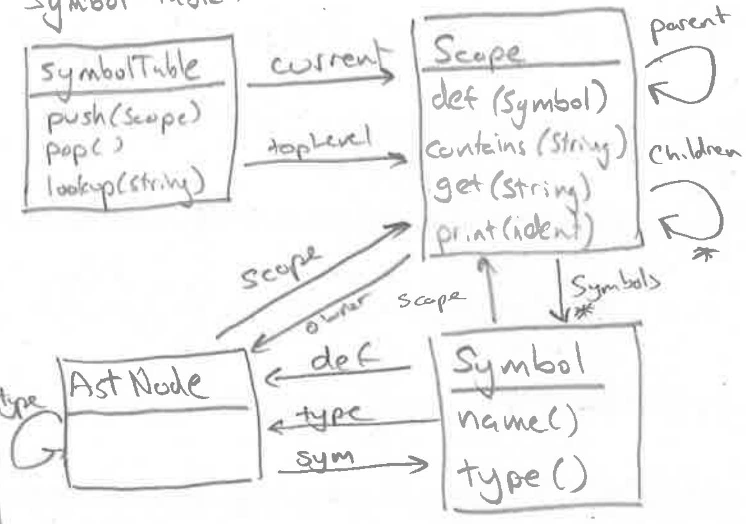
```
{ x=1;
  { x=2;
    x := x+1
  }
  print (x: string); # prints 1
}
```

Scope owner	Example
BlockStmt	{ x="hi"; print(x); }
ForStmt	for j in ["a","b"] { ... }
RecordLit	(f=1, g="two")
RecordType	(f: int, g: string)
FunDef	f = fun(p:int, q:int) { ... }

Program	f = fun ...
	main = fun ...

II Scope analyzer

Symbol table:



Translation scheme:

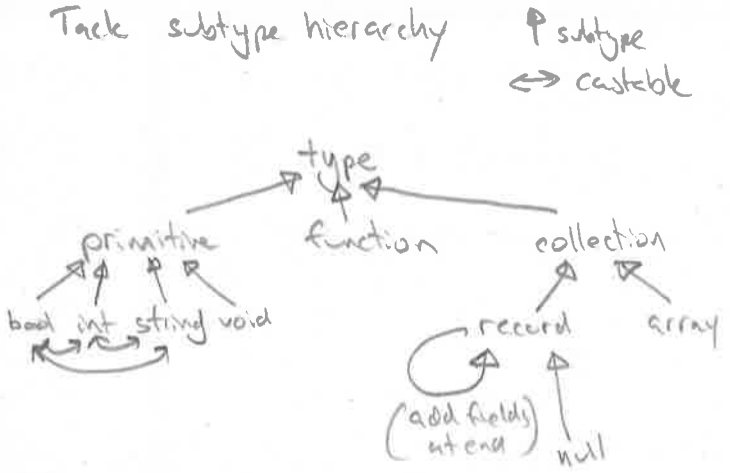
B → {	B.scope = new Scope (B, current) push (B.scope)
L	
}	pop()
L → SL | ε	
S → B | D | V	
D → id = E;	D.sym = new VarSym (D, current) current.def (D.sym) (error if duplicate)
V → id	

See also: Dragon Book Figure 2.38 (page 90)

Name and type analysis pg. 2

III Types and their relations

Tack subtype hierarchy



Example:

```
a = (x=1, y="two");
b = (x=3);
b:=a;        # subtype
a:=b :(x:int, y:string); #cast
print ("3");
print(3:string); #cast
print ("" + 3); #coercion
```

IV Type Analyzer

Translation scheme:

B → {	push(B.scope);
L	
}	pop()
L → SL | ε	
S → B|D|U	
D → id = E;	D.sym.type = E.type
U → id ;	U.sym = lookup(id) (error if not found or not var)

Example Java implementation:

See pr3.pdf, last page