Type Checking: Declarations

$$\begin{array}{llll}
T & \longrightarrow & \mathrm{int} & \{\ T.type = int;\ \} \\
T & \longrightarrow & \mathrm{float} & \{\ T.type = float;\ \} \\
D & \longrightarrow & T\ \ \mathrm{id} & \{\ D.type = T.type; \\
 & & & sym\_enter(\mathrm{id}.name,\ D.type);\ \} \\
D & \longrightarrow & D_1,\ \mathrm{id} & \{\ D.type = D_1.type; \\
 & & & sym\_enter(\mathrm{id}.name,\ D.type);\ \}
\end{array}$$

Type Checking Expressions

$E \longrightarrow$ int_const { $E.type = int;$ }

$E \longrightarrow$ float_const { $E.type = float;$ }

$E \longrightarrow$ id { $E.type = sym_lookup($id$.name,$ type$);$ }

$E \longrightarrow E_1 + E_2$ { if $(E_1.type \notin \{int, float\})$ OR

$(E_2.type \notin \{int, float\})$

$E.type = error;$

else if $E_1.type == E_2.type == int$

$E.type = int;$

else $E.type = float;$

}

int float

int

coerce into float

$\boxed{(\boxed{s + x}) + y}$

str int int

i + f

float(i) + f

Type Checking (contd.)

$E \longrightarrow \boxed{E_1\ [\ E_2\ ]}$ { if $E_1.type$ == array(**S**, **T**) AND

$E_2.type$ == int

$E.type$ = **T**

else $E.type$ = error }

$E \longrightarrow\ *\ E_1$ { if $E_1.type$ == ptr(**T**)

$E.type$ = **T**

else $E.type$ = error }

$E \longrightarrow \boxed{\&\ E_1}$ { $E.type$ = ptr($E_1.type$) }

Ref Expr

l-values

r-values

Expr

x = 3;

r-value

2 * x

x [5]

x . i = . . .

Type Checking (contd.)

(*f)

x.f

$$E \longrightarrow E_1\ E_2 \quad \{\ \text{if } E_1.type \equiv \mathbf{arrow}(\mathbf{S}, \mathbf{T}) \text{ AND } E_2.type \equiv \mathbf{S}\quad E.type = \mathbf{T} \quad \text{else} \quad E.type = \mathbf{error}\ \}$$

(fn x → x * 2)

$$E \longrightarrow (\ E_1,\ E_2\ ) \quad \{\ E.type = \mathbf{tuple}(E_1.type,\ E_2.type)\ \}$$

```
int f(int a, float b) {
        ...
}
```

f : int * float → int

Resolving Names

Overloading

static name resolution

What entity is represented by `t.area()`?

- Determine the type of `t`.

  `t` has to be of type user(c).

f(2.0, 3)

int f(int a, int b);

float f(float a, float b);

- If c has a method of name `area`, we are done.

  Otherwise, if the superclass of c has a method of name `area`, we are done.

  Otherwise, if the superclass of superclass of c...

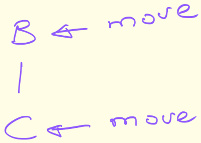
  ⟹ Determine the nearest *superclass* of class c that has a method with name `area`.

Resolving Names (contd.)

```
class Rectangle {
  int x,y; // top lh corner
  int l, w; // length and width

  Rectangle move() {
    x = x + 5;     y = y + 5;
    return this;
  }
  Rectangle move(int dx, int dy) {
    x = x + dx;    y = y + dy;
    return this;
  }
}
```

Resolving Names (contd.)



What entity is represented by `move` in `r.move(3, 10)`?

- Determine the type *C* of `r`.
- Determine the nearest *superclass* of class *C* that has a method with name `move` ***such that*** `move` ***is a method that takes two*** `int` ***parameters.***

Type Checking Statements

$S \longrightarrow id := E$ { if isSubType($E.type$, $id.type$)

$S.type$ == void

else $S.type$ = error }

$S \longrightarrow S_1;\ S_2$ { if ($S_1.type == S_2.type == void$)

$S.type$ == void

else $S.type$ = error }

$S \longrightarrow$ if E then

S_1 else S_2 { if ($S_1.type == S_2.type == void$)

&& ($E.type$ == $bool$)

$S.type$ == void

else $S.type$ = error }

$a := (b = c)$

$a = expr$