



CSCI 742 - Compiler Construction

Lecture 23

More Type Rules

Instructor: Hossein Hojjat

March 23, 2018

Recap: Type Judgments and Type Rules

$$\boxed{\Gamma \vdash e : T}$$

If the (free) variables of e have types given by the type environment Γ , then e (correctly) type checks and has type T

$$\boxed{\frac{\Gamma \vdash e_1 : T_1 \quad \dots \quad \Gamma \vdash e_n : T_n}{\Gamma \vdash e : T}}$$

If e_1 type checks in Γ and has type T_1
and ...

and e_n type checks in Γ and has type T_n
then e type checks in Γ and has type T

Type Rules for Block

$$\frac{\overbrace{\Gamma \oplus \{(x_1, T_1), \dots, (x_n, T_n)\}}^{\Gamma_1} \quad \Gamma_1 \vdash s_1 : \text{void} \quad \dots \quad \Gamma_1 \vdash s_n : \text{void}}{\Gamma \vdash \{T_1 x_1; \dots; T_n x_n; s_1; \dots; s_n\} : \text{void}}$$

Type Rules for Block

Empty:

$$\frac{}{\Gamma \vdash \{\} : \text{void}}$$

Single Statement:

$$\frac{\Gamma \vdash s : \text{void}}{\Gamma \vdash \{s\} : \text{void}}$$

$$\frac{\Gamma \oplus \{(x, T)\} \vdash \{t_2; \dots; t_n\} : \text{void}}{\Gamma \vdash \{T x ; t_2; \dots; t_n\} : \text{void}}$$

declaration is first

$$\frac{\Gamma \vdash s_1 : \text{void} \quad \Gamma \vdash \{t_2; \dots; t_n\} : \text{void}}{\Gamma \vdash \{s_1 ; t_2; \dots; t_n\} : \text{void}}$$

statement is first

Exercise

- Type check the if-expression

```
class World {
  boolean z;
  int u;
  int f(boolean y) {
    z = y;
    if (u > 0) {
      u = u - 1;
      int z;
      z = f(!y) + 3;
      u = z + z;
    }
    else {u = 0;}
    return u
  }
}
```

Exercise

```
int fact(int x) {  
    if (x == 0) return 1;  
    else return  
    return x * fact(x - 1);  
}
```

- Prove $\Gamma \vdash x * \text{fact}(x-1) : \text{int}$
where:

$\Gamma = \{(\text{fact} : \text{int} \rightarrow \text{int}), (x : \text{int})\}$

Mutual Recursion

- Example:

```
int f(int x) {  
    return g(x) + 1;  
}  
int g(int x) {  
    return f(x) - 1;  
}
```

- Need environment containing at least

$f : \text{int} \rightarrow \text{int}$ and $g : \text{int} \rightarrow \text{int}$

when checking both f and g

Two-pass approach:

- Scan top level of AST picking up all function signatures and creating an environment binding all global identifiers
- Type-check each function individually using this global environment

- Using array as an expression, on the right-hand side

$$\frac{\Gamma \vdash a : T[] \quad \Gamma \vdash i : \text{int}}{\Gamma \vdash a[i] : T}$$

- Assigning to an array

$$\frac{\Gamma \vdash a : T[] \quad \Gamma \vdash i : \text{int} \quad \Gamma \vdash e : T}{\Gamma \vdash (a[i] = e) : \text{void}}$$

- Type check the body of function

```
void next(int[] a, int k) {  
    a[k] = a[a[k]];  
}
```

Array Types

- Various kinds of array types in different programming languages

array(T)

- Array with elements of type T and no bounds
- C, Java: `int []`, Modula-3: `array of integer`

array(T, S)

- Array with size, may be indexed $0..size-1$
- C: `int [10]`, Modula-3: `array[10] of integer`

array(T, L, U)

- Array with upper/lower bounds
- Pascal or Ada: `array[2..5] of integer`

array(T, S_1, \dots, S_n)

- Multi-dimensional arrays
- FORTRAN: `real (3, 5)`