Type Checking (Part-2)

Course : 2CS701/IT794 – Compiler Construction

Prof Monika Shah

Nirma University

Ref : Ch.6 Compilers Principles, Techniques, and Tools by Alfred Aho, Ravi Sethi, and Jeffrey Ullman

Glimpse

- Quick Recap of Type Checking Part-1
- Type System (contd….)
  - Checking of Identifier Re-declaration and Un-declaration
  - Checking of Function Declaration and Invocation
  - Type Conversion and Coercion
- Implementation in YACC
  - Constructing Type Graph in Yacc
  - Type Checking in Yacc
  - Type Coercion in Yacc
  - Checking L-Values and R-Values in Yacc

Checking of Identifier Re-declaration and Un-declaration

$E \rightarrow$ **true** | **false**	{ *E*.type = *boolean* }
$E \rightarrow$ **num**	{ *E*.type = *integer* }
…	
$E \rightarrow$ **id**	{ *lookup*(**id**.entry) **if** (**id**.type = NULL) **then** print **'Identifier un-declared'** E.type = type_error **else** E.type = lookup (id.entry) }
$D \rightarrow$ **id :** *T*	{ lookup(**id.**entry) ***if (id.****type != NULL)* **then** *print 'Idenitfier re-declaration'* *D.type = type_error* ***else*** *addtype*(**id**.entry, *T*.type) }
$T \rightarrow$ **boolean**	{ *T*.type := *boolean* }
$T \rightarrow$ **char**	{ *T*.type := *char* }
…	

A Simple Language Example: Functions

$T \rightarrow T \texttt{ -> } T$

Function type declaration

$E \rightarrow E \texttt{(} E \texttt{)}$

Function call

Example:

```
v : integer;
odd : integer -> boolean;
if odd(3) then
    v := 1;
```

Simple Language Example: Function Declarations

$T \rightarrow T_1$ **->** T_2 { T.type := *function*(T_1.type, T_2.type) }

Parametric type:
type constructor

Simple Language Example: Checking Function Invocations

$E \rightarrow E_1$ **(** E_2 **)** { E.type := **if** E_1.type = *function*(s, t) **and** E_2.type = s **then** t **else** *type_error* }

Type Conversion and Coercion

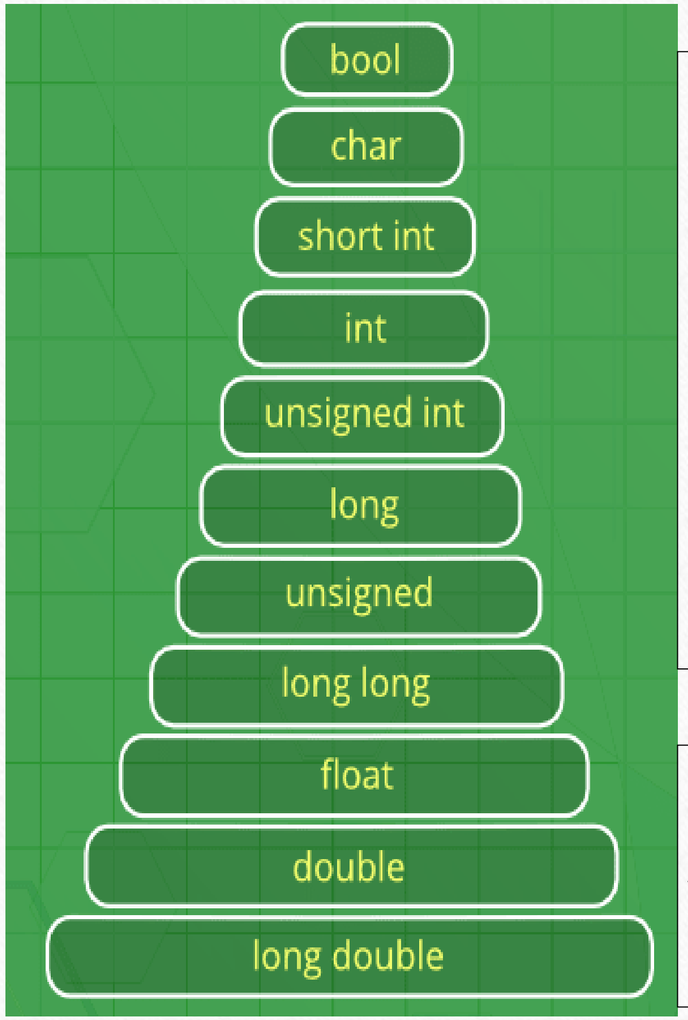
- ***Type conversion*** is **explicit**, for example using type casts

- ***Type coercion*** is **implicitly** performed by the compiler to generate code that converts types of values at runtime

- Both require a *type system* to check and infer types from (sub)expressions

Example of Type Coercion

1
2
3
4
5
6
7
8
9
10
11



1) $E \rightarrow E_1$ **op** E_2 { E.type :=

if ((E_1.type = *integer* **and** E_2.type = *float)* ***or*** (E_2.type = *integer* **and** E_1.type = *float)*) **then** *float*

else if (E_2.type = *float* **and** E_1.type = *float)*) **then** float
else if (E_2.type = *integer* **and** E_1.type = integer*)*) **then** integer

else if … **else** type_error }

2) $E \rightarrow E_1 + E_2$ { E.type := **if** ((E_1.type = *type_error* **or** E_2.type = type_error*)* ***then*** *type_error*

else max(E_2.type , E_1.type *)*}

Example of Type Conversion

$E \rightarrow$ (T) E_1 { E.type := **if** ((E_1.type = *type_error*)

then *type_error*

else *T.type)*}

Constructing Type Graphs in Yacc

`Type *mkint()` construct int node if not already constructed

`Type *mkarr(Type*,int)` construct array-of-type node if not already constructed

`Type *mkptr(Type*)` construct pointer-of-type node if not already constructed

Constructing Type Graphs in Yacc

```
%union
{ Symbol *sym;
  int num;
  Type *typ;
}
%token INT
%token <sym> ID
%token <int> NUM
%type <typ> type
%%
decl : type ID                  { addtype($2, $1); }
     | type ID '[' NUM ']'      { addtype($2, mkarr($1, $4)); }
     ;
type : INT                      { $$ = mkint(); }
     | type '*'                 { $$ = mkptr($1); }
     | /* empty */              { $$ = mkint(); }
     ;
```

Type Checking in Yacc

```
%{
enum Types {Tint, Tfloat, Tpointer, Tarray, … };
typedef struct Type
{ enum Types type;
  struct Type *child; // at most one type parameter
} Type;
%}

%union
{ Type *typ;
}

%type <typ> expr

%%
…
```

Type Checking in Yacc (contd…)

```
…
%%

expr : expr ‘+’ expr { if ($1->type != Tint
                         || $3->type != Tint)
                          semerror(“non-int operands in +”);
                       $$ = mkint();
                       emit(iadd);
                     }
```

Type Coercion in Yacc

```
…
%%
expr : expr ‘+’ expr
       { if ($1->type == Tint && $3->type == Tint)
         { $$ = mkint(); emit(iadd);
         }
         else if ($1->type == Tfloat && $3->type == Tfloat)
         { $$ = mkfloat(); emit(fadd);
         }
         else if ($1->type == Tfloat && $3->type == Tint)
         { $$ = mkfloat(); emit(i2f); emit(fadd);
         }
         else if ($1->type == Tint && $3->type == Tfloat)
         { $$ = mkfloat(); emit(swap); emit(i2f); emit(fadd);
         }
         else semerror(“type error in +”);
           $$ = mkint();
       }
```

Checking L-Values and R-Values in Yacc

```
%{
typedef struct Node
{ Type *typ;  // type structure
  int islval; // 1 if L-value
} Node;
%}

%union
{ Node *rec;
}

%type <rec> expr

%%
…
```

Checking L-Values and R-Values in Yacc (contd…)

```
expr : expr ‘+’ expr
        { if ($1->typ->type != Tint || $3->typ->type != Tint)
            semerror(“non-int operands in +”);
          $$->typ = mkint();
          $$->islval = FALSE;
          emit(…);
        }
     | expr ‘=’ expr
        { if (!$1->islval || $1->typ != $3->typ)
            semerror(“invalid assignment”);
          $$->typ = $1->typ;
          $$->islval = FALSE;
          emit(…);
        }
     | ID
        { $$->typ = lookup($1);
          $$->islval = TRUE;
          emit(…);
        }
```

Q and A