


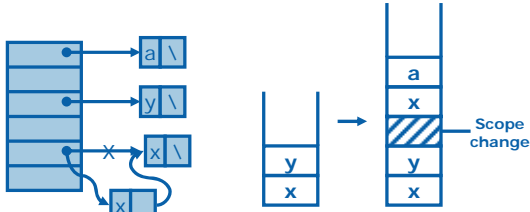
Semantic Analysis: PA5 Workshop

CS 471
October 17, 2007



Maintaining Symbol Tables

Hash table stores variable->type mappings
Scope stack used to enter/exit scope



1

Tiger Symbol Tables

Two namespaces ... two symbol tables

- **Tenv** - Namespaces for types
- **Venv** - Namespaces for vars and functions

2

Tenv: Type Environment

[See types.h]

X → types

```

Struct Ty_ty_ {
    enum{Ty_record, Ty_nil, Ty_int, Ty_string, Ty_array,
        Ty_name, Ty_void} kind;
    union {Ty_fieldList record;
        Ty_ty array;
        struct {S_symbol sym; Ty_ty ty;} name;
    } u;
};
    
```

3

Venv

[env.c: You write this ... use book and types.c as a guide]

X → **E_venventry**

```

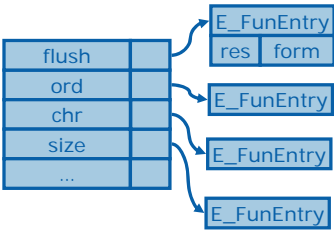
Struct E_venventry_ {
    enum{E_varEntry, E_funEntry} kind;
    union{struct {Ty_ty ty;} var;
        struct {Ty_tyList formals; Ty_ty result;} fun;
    } u;
};
    
```

E_venventry **E_VarEntry**(Ty_ty ty);
E_venventry **E_FunEntry**(Ty_tyList formals, Ty_ty result);
S_table **E_base_tenv**(void);
S_table **E_base_venv**(void);

4

Initial venv

Base functions from Appendix A



5

High-Level View of the Project

types.c
env.c
semant.c

- Type checking
- Generate intermediate code

For now (PA5), semant.c will just do type checking

- int code = NULL pointer

Later, semant.c will generate intermediate code

6 CS 471 - Fall 2007

We Will Write

Recursive functions to operate on the AST

transTy:
A_ty * tenv → Ty_ty

transVar:
A_var * tenv * venv → Ty_ty (+ int code)

transExp:
A_exp * tenv * venv → Ty_ty (+ int code)

transDec:
A_dec * tenv * venv → () /* side effect symtab */

7 CS 471 - Fall 2007

transTy – for name types

Given symbol n:

1. Lookup n in tenv
2. Return Ty_NAME
(or return result of lookup. If lookup fails return an error and assume int)

8 CS 471 - Fall 2007

transTy – for record types

Example

y: { a:x; b:y; c;z; } } Field list

| | | |
|---|--------|--|
| a | type x | |
| b | type y | |
| c | type z | |

1. Lookup x, y, z in tenv
2. Place the tyfields list in a record type and return it

9 CS 471 - Fall 2007

transTy – for array types

type t = array of tp

1. Lookup type of tp in tenv
2. Return an array type

Example:
type t = array of int

10 CS 471 - Fall 2007

For Recursive Types

type t1 = record { x: int; y: t2; }
type t2 = record { a: string; b: t1; }

1. Put all "headers" in the environment first
header: type t1 = S_enter(tenv, t1, Ty_NAME(t1, NULL))
2. Process all of the "bodies"
body: record { x: int; y:t2}

11 CS 471 - Fall 2007

We Will Write (revisited)

transTy:

`A_ty * tenv → Ty_ty`

transVar:

`A_var * tenv * venv → Ty_ty (+ int code)`

transExp:

`A_exp * tenv * venv → Ty_ty (+ int code)`

transDec:

`A_dec * tenv * venv → () /* side effect symtab */`

12

CS 471 – Fall 2007

Summary

Type checking is simple a walk of our AST

Upcoming Events:

- * PA5: Type Checking (October 26)
- * See assignment page for “early bird” bonus

Notes:

- * No class on Monday, November 19

13

CS 471 – Fall 2007