Lec 14 Structs

Struct

$e ::= \ldots \mid \text{alloc}(\tau) \mid *e \mid \text{alloc\_array}(\tau, e) \mid e_1[e_2]$

↑ can alloc struct ↑ field access equiv to this

Declaring `struct s ;`
Defining `struct s { τ1 f1 ; ... ; τn fn ; } ;`
Field access $s.\tau_i : \tau_i$

Structs are large type — unlike arrays which are ptrs

Type size restrictions these have to be small type
- return, local var, args
- assignment
- cond. expr
- equality
- expression used as statement

Static Semantics

`struct s* x = alloc(struct s)`

$\tau ::= \ldots \mid \text{struct } s$
$e ::= \ldots \mid e.f$
$d ::= \ldots \mid d.f$

$$\frac{\Gamma \vdash e : \text{struct } s \quad s.f : \tau}{e.f : \tau}$$

$e \to f \equiv (*e).f$

- Field names are in their own namespace
- Struct defined at most once
- Returning `struct s` implicitely declares s even before decl or defn
- Size only known after struct defn

Dynamic Semantic

alloc — fill fields with default value

$e.y \to \{x : v_x, y : v_y\}.y \to v_y$? not efficient

`(*p).y`
1. Get address of struct
2. Compute field offset
3. Dereference

Same as array, we track what fields are there

$e\ \{\tau_1\ f_1 ; \ldots\}.f$

For dynamic semantic, introduce & to get address

$\ldots\ e.f \triangleright k \to \ldots\ *(\&(e.f)) \triangleright k$
$\ldots\ \&(e.f) \triangleright k \to \ldots\ \&e \triangleright (\&(\_.f), k)$ ①
$\ldots\ a \triangleright (\&(\_.f), k) \to \ldots\ a + \text{offset}(s, f) \triangleright k$
if $a \neq 0$, a : struct s else mem exception

① How to get this address ?

$\ldots\ \&(*e) \triangleright k \to \ldots\ e \triangleright k$
$\ldots\ \&(e_1[e_2]) \triangleright k \to$ just find addr of array access at e_2

Unified rule for struct field / array loc assignment
; [just use address in general]

Shortcut assignment

`d += e` ⇝ `d = d + e` no longer works

```
int* x = 0 ;
*x += 1/0 ;
```
← should give div exn on CO ref compiler

② assignment
|
asnop(d, ⊙, e)
| |
① ② compute expr
get addr