Lec 19 Alias Analysis & Loop Optimisation

Observe : different temp can hold same address

Recall

$$\left.\begin{array}{l} \ell : t \leftarrow M[a] \\ \vdots \ \ t \text{ not redefined}, a \text{ not changed}, M[a] \text{ still available} \\ \ell' : t' \leftarrow M[a] \end{array}\right\} \rightarrow \left\{\begin{array}{l} \ell : t \leftarrow M[a] \\ \vdots \\ \ell' : t' \leftarrow t \end{array}\right.$$

• SSA will have those

can involve aliases

if t still defined at this point

Alias Analysis undecidable , but we approximate

Alias Analysis

- Use type — if differenty / size / offset / ... they're not alias

  class (a, τ, k) ≈ temp a contains an address derived from sth of type τ and offset k

  In c0 : different class ⇒ point to different objects

  1. Seed func params with type and offset = 0
  2. Propagate class info to other temps

$$\frac{\ell : a \leftarrow b \quad \text{class}(b, \tau, k)}{\text{class}(a, \tau, k)} \qquad \frac{\ell : a \leftarrow b + n \quad \text{class}(b, \tau, k)}{\text{class}(a, \tau, k + n)}$$

$$\frac{\ell : a \leftarrow b + t \quad \text{class}(b, \tau, k)}{\text{class}(a, \tau, \top)}$$

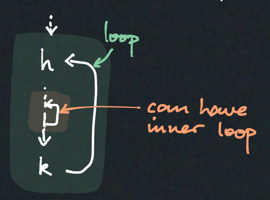
assume it can be any possible offset

Define : $\top + n = \top$, $\top + \top = \top$ — exists fancier things you can do

- Use place in the code where they're allocated

Loop Optimisation

In CFG , there's loop from h to k when h > k and ∃ edge k → h



Usually good to optimise inner loop first

- Unrolling — if know how many iterations , linearise loop
- Loop hoisting — move invariants out of loop
- Loop order interchange
- Loop inversion — put condition at end
- Loop fusion

  map g (map f L) → map (g∘f) L

- Induction variable — eg. use address as loop counter

  $i^k = k \cdot a$

Loop hoisting

```
init:            ← pre header
   i0 ← 0
   goto loop(i0)

loop(i0):        ← loop header
   t ← w * h
   if (i1 ≥ t) then exit else body(i1)

body(i2):
   t ← 4 * i2
   goto loop(i3)
```

hoisting

Define : inv (h, p) ≈ expr p is invariant in loop h

For SSA programme :

$$\frac{c \text{ constant}}{\text{inv}(h, c)} \qquad \frac{\text{def}(\ell, x) \quad \neg\text{loop}(h, \ell)}{\text{inv}(h, x)}$$

$$\frac{\text{inv}(h, e_1) \quad \text{inv}(h, e_2)}{\text{inv}(h, e_1 \oplus e_2)}$$

↑ no side effect

$$\frac{\ell : x \leftarrow p \quad \text{loop}(h, \ell) \quad \text{inv}(h, p) \quad x \text{ is not loop param}}{\text{inv}(h, x)}$$

Then move things to pre - header

Induction Variable

Basic : $x = i * c$

Derived : $x = a * i + b$