Lec 6 AST to IR tree, Static Analysis

AST ⟶ IR tree

Goals: make explicit control flow (with jump)
make eval. order explicit (isolate effectful ops)

Grammar

▸ AST

Expr e ::= $n \mid x \mid e_1 \otimes e_2 \mid e_1 \oslash e_2 \mid e_1 \,?\, e_2 \mid f(e_1, \dots, e_n)$
(⊗: no effect, ⊘: effectful, ?: comparison)
$\mid\ !e \mid e_1 \,\&\&\, e_2 \mid e_1 \,||\, e_2$

Stmts s ::= $\text{decl}(x, \tau, s) \mid \text{assign}(x, e) \mid \text{if}(e, s_1, s_2)$
$\mid\ \text{while}(e, s) \mid \text{return} \mid \text{nop} \mid \text{seq}(s_1, s_2)$

▸ IR tree — only no effect ops in pure exprs. side effect (effectful op, funcs) as commands

Pure expr p ::= $n \mid x \mid p_1 \otimes p_2$
Fun body r ::= $c_1, \dots, c_n$
Commands c ::=
- $x \leftarrow p$
- $x \leftarrow p_1 \oslash p_2$] side fx
- $x \leftarrow f(p_1, \dots, p_n)$] side fx
- $\text{if}(p_1 \,?\, p_2)$ then l_1 else l_2
- goto l
- return p
- l:

Translating integer exprs

$tr(e) = \langle ins(e), res(e) \rangle$ (ins(e): instructions of e; res(e): result of e)
$= \langle r, p \rangle$ (r: seq of commands; p: res of e as a pure expr)

$tr(n) = \langle \cdot, n \rangle$
$tr(x) = \langle \cdot, x \rangle$
$tr(e_1 \otimes e_2) = \langle ins(e_1); ins(e_2), res(e_1) \otimes res(e_2) \rangle$
$tr(e_1 \oslash e_2) = \langle ins(e_1); ins(e_2); x \leftarrow res(e_1) \oslash res(e_2), x \rangle$
$tr(f(e_1, \dots, e_n)) = \langle ins(e_1); \dots ins(e_n); x \leftarrow f(res(e_1), \dots, res(e_n)), x \rangle$

↳ all pure
↳ side fx could go here

Translating Statements

statement ⟶ seq of commands

$tr_s(\text{assign}(x, e)) = ins(e); x \leftarrow res(e)$
$tr_s(\text{return } e) = ins(e); \text{return } (res(e))$
$tr_s(\text{nop}) =$ ·
$tr_s(\text{seq}(s_1, s_2)) = tr_s(s_1); tr_s(s_2)$
$tr_s(\text{if}(e, s_1, s_2)) = ins(e); \text{if } (res(e) \neq 0) \text{ then } l_1 \text{ else } l_2$
$l_1: tr_s(s_1); \text{goto } l_3$
$l_2: tr_s(s_2); \text{goto } l_3$
$l_3:$

↳ missing short circuit
↳ to get basic block (start with label, end with jump)

$tr_s(\text{while}(e, s)) = l_1: ins(e);$
$\text{if } (res(e) \neq 0) \text{ then } l_2 \text{ else } l_3$
$l_2: tr_s(s); \text{goto } l_1$
$l_3:$

Translating Boolean Ops

Do we do $tr(b) = \langle ins(b), res(b) \rangle$?

Somewhat inefficient to translate b with some cmp then do some cmp again outside the bool (e.g. in loop condition)

Semantic: $cp(b, l, l') = r$
≈ if b then r jumps to l
else r jumps to l'

$tr_s(\text{if}(b, s_1, s_2)) =$ $cp(b, l_1, l_2);$
$l_1: tr_s(s_1); \text{goto } l_3$
$l_2: tr_s(s_2); \text{goto } l_3$
$l_3:$

$cp(e_1 \,?\, e_2, l_1, l_2) = ins(e_1); ins(e_2)$
$\text{if } (res(e_1) \,?\, res(e_2)) \text{ then } l_1 \text{ else } l_2$

$cp(!e_1, l_1, l_2) = ins(e); cp(res(e), l_2, l_1)$
$= cp(e, l_2, l_1)$

$cp(e_1 \,\&\&\, e_2, l_1, l_2) =$ $cp(e_1, l', l_2)$
$l': cp(e_2, l', l_2)$

Static Semantics

Check the programme satisfies language specification
↳ variable declared
↳ at least a return
↳ well typed
↳ etc.

Typing τ ::= int | bool

$x + 5, \ x : \text{bool} \Rightarrow \ ?$

Type judgement $\Gamma \vdash e : \tau$ "e has type τ in context Γ"

Γ = {variable ↦ type} = $x_1 : \tau_1, x_2 : \tau_2, \dots, x_n : \tau_n$

$$\frac{}{\Gamma \vdash n : \text{int}} \quad \text{↳ int const}$$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash n : \tau}$$

$$\frac{\Gamma \vdash e_1 : \tau, \ \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 = e_2 : \text{bool}}$$

Ex.

$$\frac{\dfrac{}{x : \text{int} \vdash x : \text{int}} \quad \dfrac{}{x : \text{int} \vdash 5 : \text{int}}}{x : \text{int} \vdash x = 5 : \text{bool}}$$