2/10/92 Notes on Control Flow Analysis

M. Wand

Consider untyped CBV λ-calculus with base constants only:

$$\text{Exprs } M ::= c_n \mid x \mid MM \mid \lambda x.M \mid \underline{\text{choose}}\ MM \mid g\,M$$

special form (for primitive, eg add1)

numerals

nondeterministic choice abstracts if

$$\text{Values } V ::= c_n \mid x \mid \lambda x.M$$

optional; mostly will consider closed values only.

Call-by-value rules: write $M \Rightarrow V$ for eval $(M) = V$: i.e. you don't take transitive closure

(1) $x \Rightarrow x$ (2) $\lambda x.M \Rightarrow \lambda x.M$ (3) $c_n \Rightarrow c_n$

$$(4)\quad \frac{M \Rightarrow \lambda x.P \qquad N \Rightarrow N' \qquad P[N'/x] \Rightarrow V}{MN \Rightarrow V}$$

note: if $M \Rightarrow c_n$, then there is no V s.t. $MN \Rightarrow V$.

So error is the same as non-termination.
We can fix this later.

$$(5)\quad \frac{M \Rightarrow c_n}{g\,M \Rightarrow c_{g(n)}} \qquad \text{or similar} \qquad \frac{M \Rightarrow V}{\text{choose } MN \Rightarrow V} \qquad \frac{N \Rightarrow V}{\text{choose } MN \Rightarrow V}$$

Types & Flows

type of closure

$$T ::= \underline{int} \mid (\lambda x.M)[A]$$

$$A ::= \epsilon \mid A[x:F]$$

$$F ::= \text{finite set of } T$$

A: Vars $\rightarrow F$ fv$(\lambda x.M) \rightarrow F$

σ: Vars $\rightarrow$ closed values

Define $V:T$ & $V:F$ recursively:

$$c_n : int$$

$$\sigma(x_1):A(x_1) \ldots \sigma(x_n):A(x_n) \Rightarrow (\lambda x.M\sigma) : (\lambda x.M)[A]$$

(say $\lambda x.M\sigma$ is an A-instance of $\lambda x.M$)

$$V : F \text{ iff } \exists T \in F \text{ s.t. } V:T$$

2/11/92 ②

OCFA

Given a (closed?) pgm M, would like to find the possible values of all the bound vbles. But this assumes an instrumented implementation, which is dubious.

Rather, let's try to <u>annotate</u> the program & generate verification conditions s.t. that any solution to the VC's is a valid annotation (whatever that means, ~ we'll see this later).

So associate a flow variable with every subexpression & binding instance in the pgm. We'll write ϕ_M for the flow variable associated with {a particular occurrence of} M. Then we'll generate constraints s.t. if we have any assignment to the ϕ's that satisfies the constraints, then the annotation is sound in the following sense:

<u>Thm</u> ~~If~~ $A \vdash M : \phi$, $\sigma : A$, and $M\sigma \Rightarrow V$, then $V : \phi$.

This says that if the inputs to M (via free variables) satisfy A, then the output satisfies ϕ.

VC rules:

We assign A via binding occurrences

A restricted to free vars of $\lambda x.M$

$$\frac{A[x:\phi_x] \vdash M : \phi_M}{A \vdash (\lambda x.M) : \phi_{\lambda x.M}} \qquad \{(\lambda x.M)[A \restriction fv(\lambda x.M)]\} \subseteq \phi_{\lambda x.M}$$

$$A \vdash x : \phi_x \qquad \phi_x = A(x)$$

this occurrence

$$\frac{A \vdash M : \phi_M \qquad A \vdash N : \phi_N}{} \qquad \text{if } (\lambda x.P)[B] \in \phi_M, \text{ then } \phi_N \subseteq \phi_x$$

$$\& \ \phi_P \subseteq \phi_{MN}$$

2/10/92 (3)

<u>Proof of Thm 1</u>. Proof is by induction on derivation of $M\sigma \Rightarrow V$. (Operational Semantics!)
By cases on form of M:

(1) If M is vble or λ, trivial (base step)

(2) Combination: Assume we have $A : \sigma$ + $(MN)\sigma \Rightarrow V$. $(MN)\sigma \equiv (M\sigma)(N\sigma)$, so by defn of $\Rightarrow$, must have $M\sigma \Rightarrow \lambda x.P_1$, $N\sigma \Rightarrow N_1$ + $P_1[N_1/x] \Rightarrow V$. Since these derivations are shorter, the IH applies, so $\lambda x : P_1 : \phi_M$ + $N_1 : \phi_N$. So there must be a $(\lambda x.P)[B] \in \phi_M$ s.t. $\lambda x.P_1$ is a B-instance of $\lambda x.P$, say $\lambda x.P_1 \equiv (\lambda x.P\sigma)$. Now, $N_1 : \phi_N \subseteq \phi_x$ so $P_1[N_1/x]$ is a $B[x:\phi_x]$-instance of P_1. $\therefore$ by IH at $P_1[N_1/x] \Rightarrow V$, $V : \phi_P \subseteq \phi_{MN}$. ▨

<u>Observations</u>

(1) No collecting interp, no CPS. Just "program logic" (à la Floyd-Hoare!)

(2) The constraints are closure conditions + only finitely many closure types can appear, so can solve by iteration to fixpt.

(3) These are "funny" closure conditions, since you generate new conditions (eg $\phi_N \subseteq \phi_x$, $\phi_P \subseteq \phi_{MN}$) as you go along. Can think of this as generating the data flow graph. (cf Shivers' comments on data flow vs. control flow graph. "λ: The Ultimate Computed Go-to?")

(4) Can use this to check that environments, continuations, etc are passed correctly + have only certain form, then can synthesize efficient reps for these closures

(5) What other optimizations can be justified in this way? Is there a notion of equivalence ie $M_1 \equiv M_2$ mod ϕ ?

2/10/92 (4)

Variations

1. Error semantics. Add a constant error, a type T, & a flow T (top ≡ anything) then check to see that the least soln you get isn't T.

2. Data Flow. Instead of using the lattice F, use F×D where D is your favorite lattice of assertions. Everything should still work as before. You can't abstract F very much, because then you won't have enough information to set up the data constraints. Or use CFA to set up the flow constraints (eg $\phi_N \subseteq \phi_x$) & solve those later.

3. Finer CFA variants. In OCFA, we provide a single {assertion / annotation} for every program phrase. But you might want some polymorphism e.g.

$f(n) =$ choose [] (g "a" (g n (f (n-1))))

$g(x,y) =$ Cons(x,y)

You might have 2 assertions for g, depending on what its input is. (ie where it's called from)

Pure Abstract Interpretation

$$\frac{A \vdash M : \phi_1 \qquad A \vdash N : \phi_2}{A \vdash MN : \phi}$$

if $(\lambda x.P)[B] \in \phi_1$ & $B[x:\phi_2] \vdash P : \phi_3$, then $\phi_3 \subseteq \phi$

But how to control the proliferation of subgoals?

2/10/92 (5)

1 CFA

Let C be the set of call sites in the pgm. Have a C-indexed family of assertions about each expression $A_{M,c} \vdash M : \phi_{M,c}$

$$\frac{A_{M,c} \vdash M : \phi_{M,c} \qquad A_{N,c} \vdash N : \phi_{N,c}}{A_{MN,c} \vdash^{c'} MN : \phi_{MN,c}}$$

↑
This is call site c'

$\forall y,\ A_{M,c}(y) = A_{N,c}(y) = A_{MN,c}(y)$

if $(\lambda x.P)[B] \in \phi_{M,c}$, use the c'-th assertion about $\lambda x.P$

i.e. set $\phi_{N,c} \subseteq \phi_{x,c'}$

$\phi_{P,c'} \subseteq \phi_{MN,c}$

But how to set the B's ?

$$A_{\lambda x.M,c} \vdash (\lambda x.M) : \phi_{\lambda x.M,c}$$

Need to create assertions $A_{M,c'}[x : \phi_{x,c'}] \vdash M : \phi_{M,c'}$ for all $c' \in C$.

So set $A_{\lambda x.M,c} \subseteq A_{M,c'}$

So ~~in~~ c'-th assertion about M, we merge information about the context in which $\lambda x.M$ was closed.

2/10/92 (6)

More variations

1. Let Q be a finite automaton & let $\delta: Q \times C \to Q$. Have Q-indexed family of assertions about each expression. To compute the q-th assertion about call c' use the $\delta(q, c')$-th assertion about the operator:

$$\frac{A \vdash M : \phi_{M,q} \qquad A \vdash N : \phi_{N,q}}{A \vdash c' : MN : \phi_{MN,q}}$$

if $(\lambda x.P)[B] \in \phi_{M,q}$, set

$$\phi_{N,q} \subseteq \phi_{x,\delta(q,c')}$$

$$\phi_{P,\delta(q,c')} \subseteq \phi_{MN,q}$$

2. Keep track of call site for each bound variable. This avoids merging info. in 1 CFA. So environments look like

$$[x_1 : \phi_{x_1 c_1} \cdots x_n : \phi_{x_n c_n}] \vdash M : \phi_{M, c_1 \ldots c_n}$$

meaning each x_i got its value at call site c_i. And we'll index the assertions on all of these guys. Then we have something like

$$\frac{A[x : \phi_{x,c'}] \vdash M : \phi_{M,\bar{c}c'}}{A \vdash_{\bar{c}} (\lambda x.M) : \phi_{\lambda x.M,\bar{c}}}$$

This is going to be very fine analysis.

3. Can clearly combine these ideas. This way madness lies.