Biorthogonality, Step-indexing & Compiler Correctness

Nick Benton
Microsoft Research

Chung-Kil Hur
University of Cambridge

31st Aug 2009
@ ICFP 2009

Compiler Correctness

- Type-like safety / liveness property
  - memory safety, resource usage limits, ⋯
  - most applicable in PCC scenarios

- Full Functional correctness of a compiler $\llbracket - \rrbracket$

  Compiled machine code　　　　　　　　　　Closed source program of ground type

  - $\mathrm{Obs}(\llbracket P \rrbracket) = \mathrm{Obs}(P)$: Computational adequacy
  - Compilation preserves all observational properties of source programs
  - Computational adequacy has been regarded as correctness of a compiler.

Compositional Compiler Correctness

- Problem: Computational adequacy is NOT compositional.

  $\mathbb{C}\text{-}D_1$, $\mathbb{C}\text{-}D_2$ computationally adequate

  $$\not\Rightarrow \quad Obs(Link(\mathbb{C}\,C\,D_1, \mathbb{C}\,P\,D_2)) = Obs(C[P])$$

- Solution

  ① Define $\models \subseteq MachineProg \times SourceProg$, called realizability relation

  ② $p \in MachineProg$ correctly implements $P \in SourceProg \overset{def}{\iff} p \models P$

  ③ $\mathbb{C}\text{-}D$ is correct $\overset{def}{\iff} \forall P \in SourceProg,\ \mathbb{C}\,P\,D \models P$

Requirements for $\models$

① Computational adequacy: $p \models P \Rightarrow Obs(p) = Obs(P)$

② Compositionality: $c \models C,\ p \models P \Rightarrow Link(c, p) \models C[P]$

③ Extensionality: does not over-specify intensional details

$\rightsquigarrow$ validating heavily-optimized machine code

Overview

PCF_v

$[\![-]\!]$ standard denotational semantics

ω-CPOs $\rightsquigarrow$ Advantages
$\begin{cases} \text{① Extensionality for } PCF_v \\ \text{② Equipped with a notion of approximation} \end{cases}$

$\models$ our realizability relation

= logical relation + Biorthogonality + Step-indexing

Extensionality for SECD

A notion of approximation for SECD

extended SECD

with small-step op. sem.

- All formalized and verified in Coq (using domain package by Benton et al. TPHOLs 09)

SECD machine with Eq

$$\begin{aligned}\mathit{Inst} ::= {} & \mathsf{Swap} \mid \mathsf{Dup} \mid \mathsf{PushV}\ n \mid \mathsf{Op}\ * \mid \mathsf{PushC}\ c \mid \mathsf{PushRC}\ c \\ & \mid \mathsf{App} \mid \mathsf{Ret} \mid \mathsf{Sel}(c_1, c_2) \mid \mathsf{Join} \mid \mathsf{MkPair} \mid \mathsf{Fst} \mid \mathsf{Snd} \mid \mathsf{Eq}\end{aligned}$$

Syntactic Eq test

$$\mathit{Val} ::= \underline{n} \mid \mathsf{CL}(e, c) \mid \mathsf{RCL}(e, c) \mid \mathsf{PR}(v_1, v_2)$$

$$\mathit{Config} ::= (c,\ e,\ s,\ d)$$

- c: list Inst = Code
- e: list Val = Env
- s: list Val = Stack
- d: list (Code × Env × Stack) = Dump

$$\mathit{Comp} ::= \mathit{Code} \times \mathit{Stack}$$

$$\mathit{Cont} ::= \mathit{Code} \times \mathit{Env} \times \mathit{Stack} \times \mathit{Dump}$$

$$-[=] : \mathit{Cont} \times \mathit{Comp} \to \mathit{Config} : (c, e, s, d), (c_0, s_0) \mapsto (c_0 +\!\!+\, c, e, s_0 +\!\!+\, s, d)$$

Compiler : PCF$_v$ to SECD

Values:

$$
\begin{aligned}
(\!|x_1 : t_1, \ldots, x_n : t_n \vdash x_i : t_i|\!) &= [\mathtt{PushV}\ i] \\
(\!|\Gamma \vdash \mathit{true} : \mathtt{Bool}|\!) &= [\mathtt{PushN}\ 1] \\
(\!|\Gamma \vdash \mathit{false} : \mathtt{Bool}|\!) &= [\mathtt{PushN}\ 0] \\
(\!|\Gamma \vdash n : \mathtt{Int}|\!) &= [\mathtt{PushN}\ n] \\
(\!|\Gamma \vdash \langle V_1, V_2\rangle : t_1 \times t_2|\!) &= (\!|\Gamma \vdash V_1 : t_1|\!) +\!\!+ (\!|\Gamma \vdash V_2 : t_2|\!) +\!\!+ [\mathtt{MkPair}] \\
(\!|\Gamma \vdash \text{Rec}\ f\ x \;=\; M : t \to t'|\!) &= [\mathtt{PushRC}\,((\!|\Gamma, f : t \to t', x : t \vdash M : t'|\!) +\!\!+ [\mathtt{Ret}])]
\end{aligned}
$$

Expressions:

$$
\begin{aligned}
(\!|\Gamma \vdash [V] : t|\!) &= (\!|\Gamma \vdash V : t|\!) \\
(\!|\Gamma \vdash \mathtt{let}\ x = M\ \mathtt{in}\ N : t'|\!) &= [\mathtt{PushC}\,((\!|\Gamma, x : t \vdash N : t'|\!) +\!\!+ [\mathtt{Ret}])] +\!\!+ (\!|\Gamma \vdash M : t|\!) +\!\!+ [\mathtt{App}] \\
(\!|\Gamma \vdash V_1\, V_2 : t'|\!) &= (\!|\Gamma \vdash V_1 : t \to t'|\!) +\!\!+ (\!|\Gamma \vdash V_2 : t|\!) +\!\!+ [\mathtt{App}] \\
(\!|\Gamma \vdash \mathtt{if}\ V\ \mathtt{then}\ M_1\ \mathtt{else}\ M_2 : t|\!) &= (\!|\Gamma \vdash V : \mathtt{Bool}|\!) +\!\!+ [\mathtt{Sel}\,(((\!|\Gamma \vdash M_1 : t|\!) +\!\!+ [\mathtt{Join}]), ((\!|\Gamma \vdash M_2 : t|\!) +\!\!+ [\mathtt{Join}]))] \\
(\!|\Gamma \vdash V_1 \star V_2 : \mathtt{Int}|\!) &= (\!|\Gamma \vdash V_1 : \mathtt{Int}|\!) +\!\!+ (\!|\Gamma \vdash V_2 : \mathtt{Int}|\!) +\!\!+ [\mathtt{Op}\,\star] \\
(\!|\Gamma \vdash V_1 > V_2 : \mathtt{Bool}|\!) &= (\!|\Gamma \vdash V_1 : \mathtt{Int}|\!) +\!\!+ (\!|\Gamma \vdash V_2 : \mathtt{Int}|\!) +\!\!+ [\mathtt{Op}\,(\lambda(n_1, n_2).n_1 > n_2 \supset 1 \mid 0)]
\end{aligned}
$$

Figure 3. Compiler for PCF$_v$

Biorthogonals (Krivine 1994; Pitts & Stark 1998)

$$\boxed{\text{used to achieve extensionality on the machine side}}$$

- $\perp\!\!\perp$-closure for a fixed observation $\Theta \subseteq \mathrm{Config}$

$$\{ p \in \mathrm{Comp} \mid \forall C \in \mathcal{C},\ C[p] \in \Theta \} \hookleftarrow\!\!| \ \mathcal{C} \subseteq \mathrm{Cont}$$

$$\begin{tikzcd}
\mathbb{P}(\mathrm{Comp}) \arrow[rr, bend right, "\text{Galois Connection}"'] & & \mathbb{P}(\mathrm{Cont}) \arrow[ll, bend right]
\end{tikzcd}$$

$$P \subseteq \mathrm{Comp} \mapsto \{ C \in \mathrm{Cont} \mid \forall p \in P,\ C[p] \in \Theta \}$$

Step-indices (Appel & McAllester 2001)

$\boxed{\text{A notion of approximation on the machine side}}$

- step-indices are used for inductive reasoning
- "$rec_n\ F$" : used to reason about recursive functions

  thanks to unwinding theorem

$$M[\mathrm{rec}\ F]\Downarrow \iff \exists m\ \forall n \geq m,\ M[\mathrm{rec}_n\ F]\Downarrow$$

- step indices : used to reason about recursive types, references
- We use step indices to reason about recursive functions

  because unwinding theorem NOT hold, due to the instruction Eq.

  - $P_k(C) \triangleq C$ has property P for k steps of excution
  - $P(C) \iff \forall k,\ P_k(C)$

Realizability relation

Recall

$$\begin{tikzcd}
PCF_v \arrow[dr, "{[\![-]\!]}"] & \\
& \omega\text{-CPOs} \arrow[dl, leftrightarrow, "\models"] \\
SECD &
\end{tikzcd}$$

What we want

$\rightsquigarrow$ $p \models d$: SECD computation p realizes denotation d

- We only observe divergence & termination.
- $p \models d \overset{\text{def}}{=} (p \lll d) \wedge (p \ggg d)$
- $p \lll d \overset{\text{def}}{=} \forall k,\ p \lhd^k d$ by step indexing
- $p \lhd^k d \overset{\text{def}}{=}$ logical relation + biorthogonals w.r.t. runs at least k steps
- $p \ggg d \overset{\text{def}}{=} d \in$ Ideal Closure $(\{d' \mid p \rhd d'\})$
- $p \rhd d \overset{\text{def}}{=}$ logical relation + biorthogonals w.r.t. termination

Realizability relation : $\triangleleft^{k}$

- Simple Types : $T ::= \mathrm{int} \mid \mathrm{bool} \mid T_1 \times T_2 \mid T_1 \to T_2$

- Standard denotation : $[\![\mathrm{int}]\!] = \mathbb{N}$, $[\![T_1 \to T_2]\!] = [\![T_1]\!] \longrightarrow [\![T_2]\!]_\perp$, ...

- ${\triangleleft_v}^{k}_{T} \subseteq Val \times [\![T]\!]$, ${\triangleleft_c}^{k}_{\Gamma \vdash T} \subseteq Comp \times ([\![\Gamma]\!] \longrightarrow [\![T]\!]_\perp)$

- $\underline{n} \; {\triangleleft_v}^{k}_{\mathrm{int}} \; n \in \mathbb{N}$

- $f \; {\triangleleft_v}^{k}_{T_1 \to T_2} \; df \in [\![T_1]\!] \to [\![T_2]\!]_\perp$ logical relation

  iff $\forall j \leq k \; \forall v \; {\triangleleft_v}^{j}_{T_1} dv , \; (App::nil, v::f::nil) \; {\triangleleft_c}^{j}_{\emptyset \vdash T_2} \; df(dv)$

- $p \; {\triangleleft_c}^{k}_{\Gamma \vdash T} \; dp \in [\![\Gamma]\!] \to [\![T]\!]_\perp$

  iff $\forall j \leq k \; \forall e \; {\triangleleft_e}^{j}_{\Gamma} \; de ,$ biorthogonality

  $$p \in \left(\{ v \mid v \; {\triangleleft_v}^{j}_{T} \; dp(de) \} \right)^{\perp^j_e \perp^j_e}$$

Properties of the realizability relation

- Computational adequacy

$$p \models \bot \in [\![\mathrm{int}]\!]_\bot \Rightarrow \forall cesd \in Cont,\ cesd[p] \Uparrow$$

$$p \models [n] \in [\![\mathrm{int}]\!]_\bot \Rightarrow \forall cesd \in Cont, \begin{cases} cesd[p] \Uparrow & \text{if } cesd[\underline{n}] \Uparrow \\ cesd[p] \Downarrow & \text{if } cesd[\underline{n}] \Downarrow \end{cases}$$

- Compositionality

$$\begin{matrix} (p, nil) \models df \in [\![\tau_1 \to \tau_2]\!]_\bot \\ (q, nil) \models da \in [\![\tau_1]\!]_\bot \end{matrix} \Rightarrow (p{+}{+}q{+}{+}App::nil, nil) \models df \odot da \in [\![\tau_2]\!]_\bot$$

let $f \Leftarrow df$ in
let $a \Leftarrow da$ in
$f(a)$

- Compiler Correctness

$$(\langle\!\langle t \rangle\!\rangle, nil) \models [\![\Gamma \vdash t : A]\!]$$

- Corollary

For $\emptyset \vdash t : \mathrm{int}$, $\quad t \Downarrow n \iff \langle\!\langle t \rangle\!\rangle$ converges to $\underline{n}$

Example : Hand optimization - Optimizing iteration

- PCF$_v$ term appn : (int→int)→int→int→int $\stackrel{def}{=}$ appn f n v = $f^n v$
- hand-optimized implementation of appn

appnoptcode = [pushC ...]

$\lambda f.\lambda n.\lambda v.$ if Eq(f, ⟨ $\lambda x.x$ ⟩)
then v
else ⟨ appn ⟩ f n v

syntactic Eq test

- Proposition

appnoptcode $\models$ ⟦ $\Gamma \vdash$ appn : (int→int)→int→int→int ⟧

Summary

We developed a realizability relation that can be used to verify correctnes of

① Compiler

② hand-optimized machine code.

In our approach,

③ correctness of machine code linking is guaranteed

Discussion & future work

- Discussion
  - 6000 lines in Coq
    excluding domain package, PCFv & its denotational semantics

- Future work
  - operational semantics
  - polymorphism

    See draft on authors' webpages.
  - recursive types
  - effects (references, exceptions, input & output, ...)
  - realistic assembly language