Logical Relations & Compositional Compiler Correctness

Chung-Kil Hur
University of Cambridge

joint work with Nick Benton (Microsoft Research)

14th Oct 2009

@ Seoul National University

Motivation : Software Verification

- Software verification

  : assure that software satisfies its specification

  e.g.

  - My sorting program does the sorting.
  - My program does not access unallocated memory.

- Usual situations

many verification techniques developed for

programs in high-level languages (C, Java, ML, HASKEL ...)

Correctness → } Compilation (with optimization)

: preserve all observational properties

programs in low-level languages (x86 assembly, ...)

We want to reason about ⟵ difficult to do directly.

Compiler Correctness : Traditional approach

Computational adequacy as compiler correctness

- Computational adequacy

compiled machine code   runnable closed source program of ground type

  - $Obs(⦇P⦈) = Obs(P)$
  - Compilation preserves all observational properties of source programs.
- State of the art work
  - realistic optimizing compiler for Clight (a large subset of C) [Leroy 06, 09]
  - compiler for untyped mini ML (without polymorphism, for only terminating programs) [Chlipala 10]

verified in Coq, Syntactic approach   dealing with POPLMARK challenge

Compiler Correctness : Problem

- Problem : Computational adequacy is NOT compositional.

  $\mathcal{A}\text{-}D_1$, $\mathcal{A}\text{-}D_2$ computationally adequate

  $$\not\Rightarrow \quad Obs(Link(\mathcal{A}CD_1, \mathcal{A}PD_2)) = Obs(C[P])$$

- Further we want to reason about hand-written machine code

  $$C \dashrightarrow \mathcal{A}CD \qquad \mathcal{A}\text{-}D \text{ is correct}$$

  $$P \overset{\text{correct (?)}}{\dashrightarrow} p \quad \Leftarrow \text{hand-optimized implementation of a library routine } P$$

  $$\Downarrow$$

  $$C[P] \overset{\text{correct}}{\dashrightarrow} Link(\mathcal{A}CD, p)$$

Compositional Compiler Correctness : Our approach

- Realizability relation

$$\boxed{\models \quad \subseteq \quad \text{MachineProg} \times \text{SourceProg}}$$

Define a notion of what it means for a piece of low-level code to implement, or realize, a particular high-level term.

- Requirements for $\models$

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

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

③ Extensionality: maximally permissive
includes as many implementations as possible.

- Compositional Compiler Correctness

$(\!|-|\!)$ is correct $\overset{\text{def}}{=}$ $\forall P \in \text{SourceProg},\ (\!|P|\!) \models P$

Equational reasoning on machine language

- We can use $\models$ for equational reasoning about machine code
- naive notion of contextual equivalence does not work due to Eq
- Our $\models$ gives type system & refined contextual equivalence

  m has type $T \overset{\text{def}}{\iff} \exists M : T \quad m \models_T M$

  $m \approx_{CTX} m' : T \overset{\text{def}}{\iff} \forall c : T \to \text{int},\ \text{Link}(c, m) \Downarrow \underline{n} \Leftrightarrow \text{Link}(c, m') \Downarrow \underline{n}$
- We can use $\models$ to reason about machine code because

  $$m \models_T m' \Rightarrow m \approx_{CTX} m'$$
- Here, extensionality of $\models$ is important!!

  Note that $\models \subsetneq \approx_{CTX}$ though we believe $\models$ is quite close to $\approx_{CTX}$

Overview

- PCF_v (simple types, recursion) [ICFP 09]

$\llbracket - \rrbracket$ standard denotational semantics

ω-CPOs $\leadsto$ Advantages
① Extensionality for PCF_v
② Equipped with a notion of approximation

$\models$ our realizability relation
= logical relation + Biorthogonality + Step-indexing

extended SECD
with small-step op. sem.

Extensionality for SECD
A notion of approximation for SECD

- System F with rec (simple types, recursion, polymorphism) [Submitted]

$\models$ logical relation + Biorthogonality + step-index
+ parametrized by precongruence on Sys F
+ a notion of approximation for Sys F

extended SECD

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

System F with recursion

Values:

$$\frac{}{\Theta;\Gamma,x:\tau\vdash x:\tau}\qquad\frac{}{\Theta;\Gamma\vdash b:\texttt{Bool}}\,(b\in\mathbb{B})\qquad\frac{}{\Theta;\Gamma\vdash n:\texttt{Int}}\,(n\in\mathbb{N})$$

$$\frac{\Theta;\Gamma,x:\tau\vdash M:\tau'}{\Theta;\Gamma\vdash\lambda x.\,M:\tau\to\tau'}\qquad\frac{\Theta;\Gamma,f:\tau\to\tau',x:\tau\vdash M:\tau'}{\Theta;\Gamma\vdash\mathrm{Rec}\ f\,x.\,M:\tau\to\tau'}\qquad\frac{\Theta;\Gamma\vdash V_i:\tau_i\ (i=1,2)}{\Theta;\Gamma\vdash\langle V_1,V_2\rangle:\tau_1\times\tau_2}$$

$$\frac{\Theta\vdash\Gamma\quad\Theta,X\vdash\tau\quad\Theta,X;\Gamma\vdash V:\tau}{\Theta;\Gamma\vdash\Lambda X.V:\forall X.\tau}\qquad\frac{\Theta\vdash\Gamma\quad\Theta,X\vdash\tau\quad\Theta\vdash\tau'\quad\Theta;\Gamma\vdash V:\tau[\tau'/X]}{\Theta;\Gamma\vdash\mathrm{Pack}\{\tau',V\}:\exists X.\tau}$$

Expressions:

$$\frac{\Theta;\Gamma\vdash V:\tau}{\Theta;\Gamma\vdash[V]:\tau}\qquad\frac{\Theta;\Gamma\vdash M:\tau\quad\Theta;\Gamma,x:\tau\vdash N:\tau'}{\Theta;\Gamma\vdash\texttt{let}\ x=M\ \texttt{in}\ N:\tau'}\qquad\frac{\Theta;\Gamma\vdash V_1:\tau\to\tau'\quad\Theta;\Gamma\vdash V_2:\tau}{\Theta;\Gamma\vdash V_1\,V_2:\tau'}$$

$$\frac{\Theta;\Gamma\vdash V:\texttt{Bool}\quad\Theta;\Gamma\vdash M_1:\tau\quad\Theta;\Gamma\vdash M_2:\tau}{\Theta;\Gamma\vdash\texttt{if}\ V\ \texttt{then}\ M_1\ \texttt{else}\ M_2:\tau}\qquad\frac{\Theta;\Gamma\vdash V_1:\texttt{Int}\quad\Theta;\Gamma\vdash V_2:\texttt{Int}}{\Theta;\Gamma\vdash V_1\star V_2:\texttt{Int}}$$

$$\frac{\Theta;\Gamma\vdash V_1:\texttt{Int}\quad\Theta;\Gamma\vdash V_2:\texttt{Int}}{\Theta;\Gamma\vdash V_1>V_2:\texttt{Bool}}\qquad\frac{\Theta;\Gamma\vdash V:\tau_1\times\tau_2}{\Theta;\Gamma\vdash\pi_i(V):\tau_i\ (i=1,2)}$$

$$\frac{\Theta\vdash\Gamma\quad\Theta,X\vdash\tau\quad\Theta;\Gamma\vdash V:\forall X.\tau\quad\Theta\vdash\tau'}{\Theta;\Gamma\vdash V\,\tau':\tau[\tau'/X]}$$

$$\frac{\Theta\vdash\Gamma\quad\Theta\vdash\tau\quad\Theta,X\vdash\tau'\quad\Theta;\Gamma\vdash V:\exists X.\tau'\quad\Theta,X;\Gamma,x:\tau'\vdash M:\tau}{\Theta;\Gamma\vdash\mathrm{Unpack}\,V\ \mathrm{as}\ \{X,x\}\ \mathrm{in}\,M:\tau}$$

Fig. 1. Typing rules for F_v

SECD machine with Eq

Inst := Pop | Push $\bar{v}$ | PushN $\underline{n}$ | PushE | PopE | Op $*$ | PushC c | PushRC c
| App | AppND | Ret | Sel(c_1, c_2) | SelND(c_1, c_2) | Join | MkPair | Fst | Snd | Eq | IsNum

Syntactic Eq test

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

Config := (c, e, s, d)

c: list Inst = Code

e: list Val = Env

s: list Val = stack

d: list (Code x Env x Stack) = Dump

Comp := Code x Stack

Cont := Code x Env x Stack x Dump

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

$$
\begin{array}{rcll}
\langle \texttt{Pop} :: c,\ e,\ v :: s,\ d \rangle & \mapsto & \langle c,\ e,\ s,\ d \rangle & \\
\langle \texttt{Push}\, i :: c,\ [v_1, \ldots, v_k],\ s,\ d \rangle & \mapsto & \langle c,\ [v_1, \ldots, v_k],\ v_i :: s,\ d \rangle & \\
\langle \texttt{PushE} :: c,\ e,\ v :: s,\ d \rangle & \mapsto & \langle c,\ v :: e,\ s,\ d \rangle & \\
\langle \texttt{PopE} :: c,\ v :: e,\ s,\ d \rangle & \mapsto & \langle c,\ e,\ v :: s,\ d \rangle & \\
\langle \texttt{PushN}\, n :: c,\ e,\ s,\ d \rangle & \mapsto & \langle c,\ e,\ \underline{n} :: s,\ d \rangle & \\
\langle \texttt{PushC}\, \mathit{bod} :: c,\ e,\ s,\ d \rangle & \mapsto & \langle c,\ e,\ \texttt{CL}\,(e, \mathit{bod}) :: s,\ d \rangle & \\
\langle \texttt{PushRC}\, \mathit{bod} :: c,\ e,\ s,\ d \rangle & \mapsto & \langle c,\ e,\ \texttt{RCL}\,(e, \mathit{bod}) :: s,\ d \rangle & \\
\langle \texttt{App} :: c,\ e,\ v :: \texttt{CL}\,(e', \mathit{bod}) :: s,\ d \rangle & \mapsto & \langle \mathit{bod},\ v :: e',\ [],\ (c, e, s) :: d \rangle & \\
\langle \texttt{App} :: c,\ e,\ v :: \texttt{RCL}\,(e', \mathit{bod}) :: s,\ d \rangle & \mapsto & \langle \mathit{bod},\ v :: \texttt{RCL}\,(e', \mathit{bod}) :: e',\ [],\ (c, e, s) :: d \rangle & \\
\langle \texttt{AppNoDump} :: c,\ e,\ v :: \texttt{CL}\,(e', \mathit{bod}) :: s,\ d \rangle & \mapsto & \langle \mathit{bod},\ v :: e',\ [],\ d \rangle & \\
\langle \texttt{AppNoDump} :: c,\ e,\ v :: \texttt{RCL}\,(e', \mathit{bod}) :: s,\ d \rangle & \mapsto & \langle \mathit{bod},\ v :: \texttt{RCL}\,(e', \mathit{bod}) :: e',\ [],\ d \rangle & \\
\langle \texttt{Op}\star :: c,\ e,\ \underline{n_2} :: \underline{n_1} :: s,\ d \rangle & \mapsto & \langle c,\ e,\ \underline{n_1 \star n_2} :: s,\ d \rangle & \\
\langle \texttt{Ret} :: c,\ e,\ v :: s,\ (c', e', s') :: d \rangle & \mapsto & \langle c',\ e',\ v :: s',\ d \rangle & \\
\langle \texttt{Sel}\,(c_1, c_2) :: c,\ e,\ v :: s,\ d \rangle & \mapsto & \langle c_1,\ e,\ s,\ (c, [], []) :: d \rangle & (\text{if } v \neq \underline{0}) \\
\langle \texttt{Sel}\,(c_1, c_2) :: c,\ e,\ \underline{0} :: s,\ d \rangle & \mapsto & \langle c_2,\ e,\ s,\ (c, [], []) :: d \rangle & \\
\langle \texttt{SelNoDump}\,(c_1, c_2) :: c,\ e,\ v :: s,\ d \rangle & \mapsto & \langle c_1,\ e,\ s,\ d \rangle & (\text{if } v \neq \underline{0}) \\
\langle \texttt{SelNoDump}\,(c_1, c_2) :: c,\ e,\ \underline{0} :: s,\ d \rangle & \mapsto & \langle c_2,\ e,\ s,\ d \rangle & \\
\langle \texttt{Join} :: c,\ e,\ s,\ (c', e', s') :: d \rangle & \mapsto & \langle c',\ e,\ s,\ d \rangle & \\
\langle \texttt{MkPair} :: c,\ e,\ v_1 :: v_2 :: s,\ d \rangle & \mapsto & \langle c,\ e,\ \texttt{PR}\,(v_2, v_1) :: s,\ d \rangle & \\
\langle \texttt{Fst} :: c,\ e,\ \texttt{PR}\,(v_1, v_2) :: s,\ d \rangle & \mapsto & \langle c,\ e,\ v_1 :: s,\ d \rangle & \\
\langle \texttt{Snd} :: c,\ e,\ \texttt{PR}\,(v_1, v_2) :: s,\ d \rangle & \mapsto & \langle c,\ e,\ v_2 :: s,\ d \rangle & \\
\langle \texttt{Eq} :: c,\ e,\ v_1 :: v_2 :: s,\ d \rangle & \mapsto & \langle c,\ e,\ \underline{1} :: s,\ d \rangle & (\text{if } v_1 = v_2) \\
\langle \texttt{Eq} :: c,\ e,\ v_1 :: v_2 :: s,\ d \rangle & \mapsto & \langle c,\ e,\ \underline{0} :: s,\ d \rangle & (\text{if } v_1 \neq v_2) \\
\langle \texttt{IsNum} :: c,\ e,\ \underline{n} :: s,\ d \rangle & \mapsto & \langle c,\ e,\ \underline{1} :: s,\ d \rangle & \\
\langle \texttt{IsNum} :: c,\ e,\ v :: s,\ d \rangle & \mapsto & \langle c,\ e,\ \underline{0} :: s,\ d \rangle & (\text{if } v \text{ is not } \underline{n} \text{ for any } n)
\end{array}
$$

Fig. 2. Operational Semantics of Extended SECD Machine

Problem with realizing recursion

- $\models \subseteq \text{Source} \times \text{Target}$ gives a type system to Target

$$m \in [\![T]\!] \overset{\text{def}}{\iff} \exists M : T,\ m \models_T M$$

- Consider CBV Untyped Lambda Calculus with $\equiv_\alpha$ as Target

$$\text{Val} ::= x \mid \lambda x.t \qquad \text{Term} ::= v \mid t\,s \mid u \equiv_\alpha v \mid \text{ERROR}$$

with the church encoding of

True, False, if-then-else, rec

- Theorem

For $\text{Type} ::= \text{Bool} \mid T \to T$ and $\{[\![T]\!] \subseteq \text{Term}\}_{T \in \text{Type}}$

(1) $\text{True}, \text{False} \in [\![\text{Bool}]\!]$

(2) $u \in [\![A_1 \to A_2 \to \cdots \to A_n \to B]\!] \Rightarrow \forall v_1 \in [\![A_1]\!], \cdots, v_n \in [\![A_n]\!],\ u\,v_1 \cdots v_n \not\rightsquigarrow \text{ERROR}$

(3) $(\forall v \in [\![A]\!],\ u\,v \rightsquigarrow v) \Rightarrow u \in [\![A \to A]\!]$ ⟵ Too strong extensionality

Then $\lambda f.\ \text{rec}\ f \notin [\![((\text{Bool} \to \text{Bool}) \to \text{Bool} \to \text{Bool}) \to \text{Bool} \to \text{Bool}]\!]$

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

$$M[rec\ F]\Downarrow \Leftrightarrow \exists m\ \forall n \geq m,\ M[rec_n\ F]\Downarrow$$

- 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
  - $P(c) \Leftrightarrow \forall k,\ P_k(c)$

Biorthogonals (Krivine 1994; Pitts & Stark 1998)

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

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

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

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

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

Realizability relation

Recall

$$\text{Sys F with rec} \leftrightarrow \text{SECD} \qquad \vDash \qquad \underset{\text{SECD}}{\underset{\uparrow}{p}} \vDash \underset{\text{Sys F with rec}}{\underset{\uparrow}{P}}$$

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

our novel notion

Realizability relation : $\lhd^k$

Types : $T ::= int \mid bool \mid T_1 \times T_2 \mid T_1 \to T_2 \mid \cdots$

- $\lhd_v{}^{k}_{T} \subseteq Val \times SrcVal$ $\qquad$ $\lhd_c{}^{k}_{\Gamma \vdash T} \subseteq Comp \times SrcTerm$
- $\underline{n} \ \lhd_v{}^{k}_{int} \ n \in \mathbb{N}$

Kripke logical relation

- $f \ \lhd_v{}^{k}_{T_1 \to T_2} \ F$

  iff $\forall j \le k \ \forall v \ \lhd_v{}^{j}_{T_1} V$, $(App::nil, v::f::nil) \ \lhd_c{}^{j}_{\emptyset \vdash T_2} \ F \cdot V$

- $p \ \lhd_c{}^{k}_{\Gamma \vdash T} \ P$

  biorthogonality w.r.t. runs at least j steps

  iff $\forall j \le k \ \forall e \ \lhd_e{}^{j}_{\Gamma} \ E$,

$$\begin{cases} p \in \left(\{ v \mid v \ \lhd_{v\perp}{}^{j}_{T} \ V \} \right)^{\perp^j_e \perp^j_e} & \text{if } P[E/\{x_i\}] \Downarrow V \\ \forall cesd \quad cesd[p] \Uparrow & \text{if } P[E/\{x_i\}] \Uparrow \end{cases}$$

Properties of the realizability relation

- Computational adequacy

$$p \models_{int} P \ \wedge\ P \Uparrow \ \Rightarrow\ \forall cesd \in Cont,\ \ cesd[p] \Uparrow$$

$$p \models_{int} P \ \wedge\ P \Downarrow \underline{n} \ \Rightarrow\ \forall cesd \in Cont,\ \begin{cases} cesd[p] \Uparrow & \text{if } cesd[\underline{n}] \Uparrow \\ cesd[p] \Downarrow & \text{if } cesd[\underline{n}] \Downarrow \end{cases}$$

- Compositionality

$$\begin{array}{l} (p, nil) \models_{T_1 \to T_2} P \\ (q, nil) \models_{T_1} Q \end{array} \ \Rightarrow\ (p \mathbin{++} q \mathbin{++} App::nil,\ nil) \models_{T_2} P.Q$$

Optimizing Compiler: Sys F to SECD

Values:

$$
\begin{aligned}
(\!|\Theta; x_1:\tau_1,\ldots,x_n:\tau_n \vdash x_i:\tau_i|\!) &= [\mathtt{Push}\ i]\\
(\!|\Theta;\Gamma \vdash \mathit{true}:\mathtt{Bool}|\!) &= [\mathtt{PushN}\ 1]\\
(\!|\Theta;\Gamma \vdash \mathit{false}:\mathtt{Bool}|\!) &= [\mathtt{PushN}\ 0]\\
(\!|\Theta;\Gamma \vdash n:\mathtt{Int}|\!) &= [\mathtt{PushN}\ n]\\
(\!|\Theta;\Gamma \vdash \langle V_1,V_2\rangle:\tau_1\times\tau_2|\!) &= (\!|\Theta;\Gamma\vdash V_1:\tau_1|\!) +\!\!+ (\!|\Theta;\Gamma\vdash V_2:\tau_2|\!) +\!\!+ [\mathtt{MkPair}]\\
(\!|\Theta;\Gamma \vdash \lambda x.\,M:\tau\to\tau'|\!) &= [\mathtt{PushC}\,((\!|\Theta;\Gamma,x:\tau\vdash M:\tau'|\!)_{\mathsf{true}})]\\
(\!|\Theta;\Gamma \vdash \mathrm{Rec}\ f\,x\ =\ M:\tau\to\tau'|\!) &= [\mathtt{PushRC}\,((\!|\Theta;\Gamma,f:\tau\to\tau',x:\tau\vdash M:\tau'|\!)_{\mathsf{true}})]\\
(\!|\Theta;\Gamma \vdash \Lambda X.\,V:\forall X.\tau|\!) &= (\!|\Theta,X;\Gamma\vdash V:\tau|\!)\\
(\!|\Theta;\Gamma \vdash \mathrm{Pack}\{\tau',V\}:\exists X.\tau|\!) &= (\!|\Theta;\Gamma\vdash V:\tau[\tau'/X]|\!)
\end{aligned}
$$

Expressions:

$$
\begin{aligned}
(\!|\mathit{ret}|\!) &= \text{if}\ \ \mathit{ret} = \text{true}\ \ \text{then}\ \ [\mathtt{Ret}]\ \ \text{else}\ \ []\\
(\!|\Theta;\Gamma\vdash[V]:t|\!)_{\mathit{ret}} &= (\!|\Theta;\Gamma\vdash V:t|\!) +\!\!+ (\!|\mathit{ret}|\!)\\
(\!|\Theta;\Gamma\vdash V_1\star V_2:\mathtt{Int}|\!)_{\mathit{ret}} &= (\!|\Theta;\Gamma\vdash V_1:\mathtt{Int}|\!) +\!\!+ (\!|\Theta;\Gamma\vdash V_2:\mathtt{Int}|\!) +\!\!+ [\mathtt{Op}\,\star] +\!\!+ (\!|\mathit{ret}|\!)\\
(\!|\Theta;\Gamma\vdash V_1 > V_2:\mathtt{Bool}|\!)_{\mathit{ret}} &= (\!|\Theta;\Gamma\vdash V_1:\mathtt{Int}|\!) +\!\!+ (\!|\Theta;\Gamma\vdash V_2:\mathtt{Int}|\!) +\!\!+\\
&\qquad [\mathtt{Op}\ (\lambda(n_1,n_2).n_1 > n_2 \supset 1 \mid 0)] +\!\!+ (\!|\mathit{ret}|\!)\\
(\!|\Theta;\Gamma\vdash\pi_1(V):\tau_1|\!)_{\mathit{ret}} &= (\!|\Theta;\Gamma\vdash V:\tau_1\times\tau_2|\!) +\!\!+ [\mathtt{Fst}] +\!\!+ (\!|\mathit{ret}|\!)\\
(\!|\Theta;\Gamma\vdash\pi_2(V):\tau_2|\!)_{\mathit{ret}} &= (\!|\Theta;\Gamma\vdash V:\tau_1\times\tau_2|\!) +\!\!+ [\mathtt{Snd}] +\!\!+ (\!|\mathit{ret}|\!)\\
(\!|\Theta;\Gamma\vdash\mathtt{if}\ V\ \mathtt{then}\ M_1\ \mathtt{else}\ M_2:\tau|\!)_{\mathsf{true}} &= (\!|\Theta;\Gamma\vdash V:\mathtt{Bool}|\!) +\!\!+\\
&\qquad [\mathtt{SelNoDump}\ ((\!|\Theta;\Gamma\vdash M_1:\tau|\!)_{\mathsf{true}}, (\!|\Theta;\Gamma\vdash M_2:\tau|\!)_{\mathsf{true}})]\\
(\!|\Theta;\Gamma\vdash\mathtt{if}\ V\ \mathtt{then}\ M_1\ \mathtt{else}\ M_2:\tau|\!)_{\mathsf{false}} &= (\!|\Theta;\Gamma\vdash V:\mathtt{Bool}|\!) +\!\!+\\
&\qquad [\mathtt{Sel}\ ((\!|\Theta;\Gamma\vdash M_1:\tau|\!)_{\mathsf{false}} +\!\!+ [\mathtt{Join}], (\!|\Theta;\Gamma\vdash M_2:\tau|\!)_{\mathsf{false}} +\!\!+ [\mathtt{Join}])]\\
(\!|\Theta;\Gamma\vdash\mathtt{let}\ x = M\ \mathtt{in}\ N:\tau'|\!)_{\mathit{ret}} &= (\!|\Theta;\Gamma\vdash M:\tau|\!)_{\mathsf{false}} +\!\!+\\
&\qquad [\mathtt{PushE}] +\!\!+ (\!|\Theta;\Gamma,x:\tau\vdash N:\tau'|\!)_{\mathit{ret}} +\!\!+ [\mathtt{PopE}]\\
(\!|\Theta;\Gamma\vdash V_1\,V_2:\tau'|\!)_{\mathsf{true}} &= (\!|\Theta;\Gamma\vdash V_1:\tau\to\tau'|\!) +\!\!+ (\!|\Theta;\Gamma\vdash V_2:\tau|\!) +\!\!+ [\mathtt{AppNoDump}]\\
(\!|\Theta;\Gamma\vdash V_1\,V_2:\tau'|\!)_{\mathsf{false}} &= (\!|\Theta;\Gamma\vdash V_1:\tau\to\tau'|\!) +\!\!+ (\!|\Theta;\Gamma\vdash V_2:\tau|\!) +\!\!+ [\mathtt{App}]\\
(\!|\Theta;\Gamma\vdash V\,\tau':\tau[\tau'/X]|\!)_{\mathit{ret}} &= (\!|\Theta;\Gamma\vdash V:\forall X.\tau|\!) +\!\!+ (\!|\mathit{ret}|\!)\\
(\!|\Theta;\Gamma\vdash \mathrm{Unpack}\,V\ \mathrm{as}\,\{X,x\}\,\mathrm{in}\,M:\tau|\!)_{\mathit{ret}} &= (\!|\Theta;\Gamma\vdash V:\exists X.\tau'|\!) +\!\!+\\
&\qquad [\mathtt{PushE}] +\!\!+ (\!|\Theta,X;\Gamma,x:\tau'\vdash M:\tau|\!)_{\mathit{ret}} +\!\!+ [\mathtt{PopE}]
\end{aligned}
$$

Fig. 3. Compiler for F_v

Compiler Correctness

- Compiler Correctness

  $(\langle t \rangle, \mathrm{nil}) \vDash_A [\![\Gamma \vdash t : A]\!]$

- Corollary

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

Example : Hand optimization - Optimizing iteration

- $\mathrm{PCF_V}$ term $\mathrm{appn} : (\mathrm{int} \to \mathrm{int}) \to \mathrm{int} \to \mathrm{int} \to \mathrm{int}$ $\stackrel{\mathrm{def}}{=}$ $\mathrm{appn}\ f\ n\ v = f^n v$

- hand-optimized implementation of appn

$$\mathrm{appnoptcode} = [\mathrm{pushC} \ \ldots]$$

$$\lambda f.\lambda n.\lambda v.\ \mathrm{if}\ \mathrm{Eq}(f, \llcorner \lambda x.x \lrcorner)\ \mathrm{then}\ v\ \mathrm{else}\ \llcorner \mathrm{appn} \lrcorner\ f\ n\ v$$

syntactic Eq test (using Eq command)

- Proposition

$$(\mathrm{appnoptcode}, \mathrm{nil}) \models_{(\mathrm{int}\to\mathrm{int})\to\mathrm{int}\to\mathrm{int}\to\mathrm{int}} [\![\Gamma \vdash \mathrm{appn} : (\mathrm{int}\to\mathrm{int})\to\mathrm{int}\to\mathrm{int}\to\mathrm{int}]\!]$$

Example : Fixpoint Combinator

$$\mathrm{FixC} = \Lambda X.\Lambda Y.\Lambda F:(X\to Y)\to X\to Y\ .\ \mathrm{Rec}\ f\ x.\ F\ f\ x$$

$$\mathrm{Ycombinator} := [\mathrm{Push}\ C \cdots\ ]$$

$\uparrow$

encoding of $\lambda F.\lambda x.(\lambda y.F(\lambda z.yyz))(\lambda y.F(\lambda z.yyz))$

- Proposition

$$(\mathrm{Ycombinator}, \mathrm{nil}) \models \underset{\forall X.\forall Y.((X\to Y)\to X\to Y)\to X\to Y}{\mathrm{Fix}\ C}$$

Example : Polymorphic List Module

$$\mathrm{SigPolList} := \forall X.\ \exists LX.\ LX \times (X \times LX \to LX) \times (LX \to \mathrm{Option}\ (X \times LX))$$

- Implementation in Source

$$\mathrm{List}\ \tau := \forall Y.\ Y \times (\tau \times Y \to Y) \to Y$$

$\mathrm{nil}_\tau := \cdots \quad : \mathrm{List}\ \tau$

$\mathrm{cons}_\tau := \cdots \quad : \tau \times \mathrm{List}\ \tau \to \mathrm{List}\ \tau$

$\mathrm{split}_\tau := \cdots \quad : \mathrm{List}\ \tau \to \mathrm{option}\ (\tau \times \mathrm{List}\ \tau)$

very inefficient (split is in $\Theta(n)$), but best implementation due to lack of recursive types.

$$\mathrm{LST} := \Lambda X.\ \mathrm{Pack}\ \{\mathrm{List}\ X,\ (\mathrm{nil}_X, \mathrm{cons}_X, \mathrm{split}_X)\} \quad : \mathrm{SigPolList}$$

- Heavy-optimized implementation in SECD

$$\mathrm{encode}\ vs = \begin{cases} 0 & \text{if } vs = [\,] \\ 2^n \times 3^m & \text{if } vs = \underline{n} :: tl \ \wedge\ \mathrm{encode}\ tl = \underline{m} \\ \mathrm{PR}(hd, \mathrm{encode}\ tl) & \text{otherwise } (vs = hd :: tl) \end{cases}$$

using IsNum command

$\mathrm{NIL} = \underset{\sim}{0} \quad \mathrm{CONS} = [\mathrm{Push}\ C \cdots] \quad \mathrm{SPLIT} = [\mathrm{Push}\ C \cdots]$

- Proposition : $(\mathrm{NIL} +\!\!+ \mathrm{CONS} +\!\!+ [\mathrm{MkPair}] +\!\!+ \mathrm{SPLIT} +\!\!+ [\mathrm{MkPair}], \mathrm{nil}) \models_{\mathrm{SigPolList}} \mathrm{LST}$

Summary

- Common Scenario

library routines → $L \dashrightarrow \ell$ ← hand-written implementation

$L' \dashrightarrow \llbracket L' \rrbracket_1$ ← $\llbracket - \rrbracket_1$ is a compiler

user program → $P \dashrightarrow \llbracket P \rrbracket_2$ ← $\llbracket - \rrbracket_2$ is another compiler

by composing → $P(L, L') \dashrightarrow \mathrm{Link}(\llbracket P \rrbracket_2, \ell, \llbracket L' \rrbracket_1)$ ← by linking

① the library implementor shows that $L \models \ell$

② The compiler implementors show that $\forall M \quad M \models \llbracket M \rrbracket_1$

$\forall M \quad M \models \llbracket M \rrbracket_2$

by compositionality of $\models$

Then it is guarnteed that $P(L, L') \models \mathrm{Link}(\llbracket P \rrbracket_2, \ell, \llbracket L' \rrbracket_1)$

by computational adequacy of $\models$ thus $\mathrm{Obs}(P(L, L')) = \mathrm{Obs}(\mathrm{Link}(\llbracket P \rrbracket_2, \ell, \llbracket L' \rrbracket_1))$

Discussion & future work

- Discussion
  - 12000 lines in Coq ← Strongly typed representation (to be submitted with Nick Benton & Andrew Kennedy
  - first compiler correctness result for Polymorphic Language!
- Future work
  - recursive types
  - effects (references, exceptions, input & output, ...)
  - realistic assembly language
  - full extensionallity (i.e. $\models \; = \; \approx_{CTX}$)
- Related work

  Recent draft of Adam Chlipala (Oct 2009) proposes Syntactic Compositional Compiler Correctness.

  ↳ now computational adequacy + compositionality → problem with polymorphism (parametricity)

  ↳ simple, easy to implement, but far from extensionality.