The Second ASIAN Symposium on Programming Languages and Systems

A Type Theory for Krivine-style Evaluation and Compilation

Kwanghoon Choi   Atsushi Ohori

JAIST

November 5, 2004

Introduction

Krivine-style evaluation: Krivine (1985)

$$
\begin{aligned}
& (\mathcal{E},\ \ \mathcal{S},\ \ (\lambda x.M)\ N_1\ N_2\ \ldots\ N_n) \\
\stackrel{*}{\Longrightarrow}\ & (\mathcal{E},\ \ V_1 \cdot V_2 \cdot \ldots \cdot V_n \cdot \mathcal{S},\ \ \lambda x.M) \\
\Longrightarrow\ & (\mathcal{E}\{x : V_1\},\ \ V_2 \cdot \ldots \cdot V_n \cdot \mathcal{S},\ \ M)
\end{aligned}
$$

Lambda abstraction is regarded either as

- a code to pop an argument or as
- a data value (or closure) to return

Application context (or “spine”)

Spine stack

Goal

A type theoretical account for Krivine-style semantics

- directly describing the static property of spine stack of the semantics
- analyzing the static structure of terms w.r.t. the semantics
- providing a basis for typed compilation based on the semantics

Our Contribution: A Krivine Type Theory

Type systems for

- a term language based on Krivine-style evaluation and
- two Krivine-style abstract machines.

Typed compilations of

- the term language into each abstract machine code language

Establishment of

- type soundness of the type systems and
- type (and semantic) correctness of the compilations

Plan

- Introduction
- √ A Typed Krivine-style Term Calculus
- A Krivine Machine and Compilation
- A Dynamically Typed Krivine Machine
- Related Work and Conclusion

A Typed Krivine-style Term Calculus

Terms and types

$$
\begin{array}{lll}
M & ::= & x \mid \lambda x.M \mid M\ M \mid \ldots \\
\tau & ::= & b \mid \Delta \to \tau \\
\Delta & ::= & \{\tau_1 \cdot \ldots \cdot \tau_n\} \\
\Gamma & ::= & \{x_1 : \tau_1,\ \ldots,\ x_n : \tau_n\}
\end{array}
$$

A typing judgment of the form

$$\Gamma \mid \Delta \rhd M : \tau$$

- M has type τ under a typing environment Γ and a (spine) stack type Δ

A Type System for Krivine-style Term Calculus

(var) $$\Gamma\{x:\tau\}\,|\,\emptyset \rhd x:\tau$$

(app) $$\frac{\Gamma\,|\,\tau_2\cdot\Delta \rhd M_1:\tau_1 \qquad \Gamma\,|\,\emptyset \rhd M_2:\tau_2}{\Gamma\,|\,\Delta \rhd M_1\ M_2:\tau_1}$$

(abs) $$\frac{\Gamma\{x:\tau_1\}\,|\,\Delta \rhd M:\tau_2}{\Gamma\,|\,\tau_1\cdot\Delta \rhd \lambda x.M:\tau_2}$$

(closure) $$\frac{\Gamma\,|\,\Delta \rhd \lambda x.M:\tau}{\Gamma\,|\,\emptyset \rhd \lambda x.M:\Delta\rightarrow\tau}$$

(install) $$\frac{\Gamma\,|\,\emptyset \rhd M:\Delta\rightarrow\tau}{\Gamma\,|\,\Delta \rhd M:\tau}$$

Properties of the Krivine Type system

Typeability

- If $\Gamma \rhd M : \tau$ then $\Gamma \,|\, \emptyset \rhd M : \tau$
- If $\Gamma \,|\, \Delta \rhd M : \tau$ then $\Gamma \rhd M : \overline{\Delta \to \tau}$

Type soundness

- If $\emptyset \,|\, \emptyset \rhd M : \tau$ and $\emptyset, \emptyset, \text{retCont} \vdash M \Downarrow V$
  then $\models V : \tau$

Plan

- Introduction
- A Typed Krivine-style Term Calculus
- √ A Krivine Machine and Compilation
- A Dynamically Typed Krivine Machine
- Related Work and Conclusion

A Krivine Machine

States: (E, S, L, C, D)

- an environment E, a spine stack S, a local stack L, a code C, and a dump stack D
- $C \;::=\; \emptyset \mid I \cdot C$
- $I \;::=\;$ Grab(x) $\mid$ MkCls(C) $\mid$ Install $\mid$ Return $\mid$ …
- $D \;::=\; \emptyset \mid (E, L, C) \cdot D$

State transitions: $(E, S, L, C, D) \longrightarrow (E', S', L', C', D')$

A Type System for Krivine Machine

A typing judgment of the form

$$\Gamma \,|\, \Delta \,|\, \Pi \triangleright C : \tau$$

cf. $(E,\ S,\ L,\ C,\ D)$

A typing rule for each instruction I

$$\text{(Rule-}I\text{)}\ \frac{\Gamma' \,|\, \Delta' \,|\, \Pi' \triangleright C : \tau}{\Gamma \,|\, \Delta \,|\, \Pi \triangleright I \cdot C : \tau}$$

cf. $(E,\ S,\ L,\ I \cdot C,\ D) \longrightarrow (E',\ S',\ L',\ C,\ D)$

A Type System for Krivine Machine (cont.)

$$\boxed{\Gamma \mid \Delta \mid \Pi \rhd C : \tau}$$

(Grab)
$$\frac{\Gamma\{x : \tau\} \mid \Delta \mid \Pi \rhd C : \tau_0}{\Gamma \mid \tau \cdot \Delta \mid \Pi \rhd \textsf{Grab(x)} \cdot C : \tau_0}$$

(Closure)
$$\frac{\Gamma \mid \Delta_0 \mid \emptyset \rhd C_0 : \tau_0 \quad \Gamma \mid \Delta \mid \Delta_0 \rightarrow \tau_0 \cdot \Pi \rhd C : \tau}{\Gamma \mid \Delta \mid \Pi \rhd \textsf{MkCls}(C_0) \cdot C : \tau}$$

(Install)
$$\frac{\Gamma \mid \Delta_2 \mid \tau \cdot \Pi \rhd C : \tau_0}{\Gamma \mid \Delta_1 \cdot \Delta_2 \mid \Delta_1 \rightarrow \tau \cdot \Pi \rhd \textsf{Install} \cdot C : \tau_0}$$

(Return)
$$\Gamma \mid \emptyset \mid \tau \rhd \textsf{Return} : \tau$$

Property of Krivine Machine Type System

Type soundness

- If $\emptyset \,|\, \emptyset \,|\, \emptyset \rhd C : \tau$ and $(\emptyset, \emptyset, \emptyset, C, \emptyset) \longrightarrow^* v$
  then $\models v : \tau$

A Type-directed Compilation for Krivine Machine

$$\boxed{\Gamma \,|\, \Delta \rhd M \leadsto_k C}$$

(abs)

$$\frac{\Gamma\{x : \tau\} \,|\, \Delta \rhd M \leadsto_k C}{\Gamma \,|\, \tau \cdot \Delta \rhd \lambda x.M \leadsto_k \text{Grab}(x) \cdot C}$$

(val)

$$\frac{\Gamma \,|\, \Delta \rhd \lambda x.M \leadsto_k C}{\Gamma \,|\, \emptyset \rhd \lambda x.M \leadsto_k \text{MkCls}(C \cdot \text{Return})}$$

(code)

$$\frac{\Gamma \,|\, \emptyset \rhd M \leadsto_k C}{\Gamma \,|\, \Delta \rhd M \leadsto_k C \cdot \text{Install}}$$

Properties of Compilation for Krivine Machine

Preservation of typing

- If $\emptyset \,|\, \emptyset \rhd M : \tau$ and $\emptyset \,|\, \emptyset \rhd M \leadsto_k C$
  then $\emptyset \,|\, \emptyset \,|\, \emptyset \rhd C \cdot \mathsf{Return} : \tau$

Semantic correctness of compiled codes

- Suppose $\emptyset \,|\, \emptyset \rhd M : \tau$ and $\emptyset \,|\, \emptyset \rhd M \leadsto_k C$.

  If $\emptyset, \emptyset, \mathsf{retCont} \vdash M \Downarrow V$
  then $(\emptyset, \emptyset, \emptyset, C \cdot \mathsf{Return}, \emptyset) \longrightarrow^* v$ s.t. $\models V \sim v : \tau$.

Plan

- Introduction
- A Typed Krivine-style Term Calculus
- A Krivine Machine and Compilation
- √ A Dynamically Typed Krivine Machine
- Related Work and Conclusion

A Dynamically Typed Krivine Machine

ZINC machine by Leroy (1992)

States: (E, S, L, C, D)

- an environment E, a spine stack S, a local stack L, a code C, and a dump stack D
- $C \ ::= \emptyset \mid I \cdot C$
- $I \ ::=$ Grab(x) $\mid$ Reduce(C) $\mid$ Push $\mid$ Return $\mid$ …
- $D \ ::= \ \emptyset \mid (E, L, S, C) \cdot D$

State transitions: $(E, S, L, C, D) \longrightarrow (E', S', L', C', D')$

A Type System for ZINC Machine

$$\boxed{\Gamma \,|\, \Delta \,|\, \Pi \rhd C : \tau}$$

(GrabAbs)

$$\frac{\Gamma\{x : \tau\} \,|\, \Delta \,|\, \emptyset \rhd C : \tau'}{\Gamma \,|\, \tau \cdot \Delta \,|\, \emptyset \rhd \mathsf{Grab}(x) \cdot C : \tau'}$$

(GrabClo)

$$\frac{\Gamma \,|\, \Delta \,|\, \emptyset \rhd \mathsf{Grab}(x) \cdot C : \tau}{\Gamma \,|\, \emptyset \,|\, \emptyset \rhd \mathsf{Grab}(x) \cdot C : \Delta \to \tau}$$

(ReturnIns)

$$\frac{\Gamma \,|\, \Delta' \,|\, \tau' \rhd \mathsf{Return} : \tau}{\Gamma \,|\, \Delta \cdot \Delta' \,|\, \Delta \to \tau' \rhd \mathsf{Return} : \tau}$$

(ReturnRet)

$$\Gamma \,|\, \emptyset \,|\, \tau \rhd \mathsf{Return} : \tau$$

Properties of ZINC Type System

Polymorphic arity of ZINC codes

- If $\Gamma \,|\, \Delta_1 \,|\, \Pi \rhd C : \Delta_2 \to \tau$
  then $\Gamma \,|\, \Delta_1 \cdot \Delta_2 \,|\, \Pi \rhd C : \tau$

Type soundness

- If $\emptyset \,|\, \emptyset \,|\, \emptyset \rhd C : \tau$ and $(\emptyset, \emptyset, \emptyset, C, \emptyset) \longrightarrow^* v$
  then $\models v : \tau$

A Type-preserving Compilation for ZINC Machine

$$\boxed{\Gamma \,|\, \Delta \rhd M \leadsto_k C}$$

(abs)
$$\frac{\Gamma\{x : \tau\} \,|\, \Delta \rhd M \leadsto_z C}{\Gamma \,|\, \tau \cdot \Delta \rhd \lambda x.M \leadsto_z \textsf{Grab}(x) \cdot C}$$

(val)
$$\frac{\Gamma \,|\, \Delta \rhd \lambda x.M \leadsto_z C}{\Gamma \,|\, \emptyset \rhd \lambda x.M \leadsto_z C}$$

(code)
$$\frac{\Gamma \,|\, \emptyset \rhd M \leadsto_z C}{\Gamma \,|\, \Delta \rhd M \leadsto_z C}$$

Properties of Compilation for ZINC Machine

Preservation of typing

- If $\emptyset \,|\, \emptyset \triangleright M : \tau$ and $\emptyset \,|\, \emptyset \triangleright M \rightsquigarrow_z C$
  then $\emptyset \,|\, \emptyset \,|\, \emptyset \triangleright C : \tau$

Semantic correctness of compiled codes

- Suppose $\emptyset \,|\, \emptyset \triangleright M : \tau$ and $\emptyset \,|\, \emptyset \triangleright M \rightsquigarrow_z C$.

  If $\emptyset, \emptyset, \mathsf{retCont} \vdash M \Downarrow V$
  then $(\emptyset, \emptyset, \emptyset, C, \emptyset) \longrightarrow^* v$ s.t. $\models V \sim v : \tau$

Comparison of Krivine and ZINC Abstract Machines

$$\boxed{(\lambda f.(\lambda x.\lambda y.M)\ (f\ 1\ 2)\ (f\ 3))\ (\lambda w.\lambda z.N)}$$

$$\lambda w.\lambda z.N \quad : \quad \{int\} \rightarrow \{int\} \rightarrow int$$

$$\lambda x.\lambda y.M \quad : \quad \{int \cdot \{int\}{\rightarrow}int\} \rightarrow int$$

	unnecessary closure	argument check
Krivine	sometimes	never
ZINC	never	always

Plan

- Introduction
- A Typed Krivine-style Term Calculus
- A Krivine Machine and Compilation
- A Dynamically Typed Krivine Machine
- √ Related Work and Conclusion

Related Work

“Krivine-style” abstract machines

- Johnsson (1984)
- Fairbairn and Wray (1987)
- Leroy (1992)
- Peyton Jones (1992)

Type systems for low-level codes

- Morrisett et al. (1998)

UnCurrying transformation

- Hannan and Hicks (1998)

Conclusion

A type theoretical framework for Krivine-style evaluation and compilation

- Type systems for a term language and two abstr. machines
- Type soundness properties

- Compilation algorithms for the abstract machines
- Type correctness and semantic correctness properties