Abstracting Abstract Machines

David Van Horn
Northeastern University
dvanhorn@ccs.neu.edu

Matthew Might
University of Utah
might@cs.utah.edu

New England Programming Languages Symposium Yale University, April 29, 2010

What is an Abstract Machine?

An idealized, low-level model of an interpreter.

Typically: a first-order state transition system

$$S \longmapsto S'$$

where each transition is a unit-cost operation.

Examples: Landin's SECD, Felleisen & Friedman's CEK, Krivine's machine, ...

Non-examples: compositional evaluation function.

What is an Abstract Interpreter?

A computable approximation to an interpreter.

Typically: a sound approximation of intensional properties of program execution.

Examples: Shivers' kCFA, ...

Sort of non-examples: constraint-based analyses, type inference, ...

Expressions Expr $e ::= x \mid \lambda x.e \mid ee$

Values Den $d ::= \langle \lambda x.e, \rho \rangle$

Continuations Cont $\kappa ::= mt \mid ar(e, \rho, \kappa) \mid fn(d, \kappa)$

Continuations represent evaluation contexts (inside out)

$E ::= [] \mid (E\ e) \mid (d\ E)$

$[] \approx mt$

$E[([]\ e)] \approx ar(e, \rho, \kappa)$

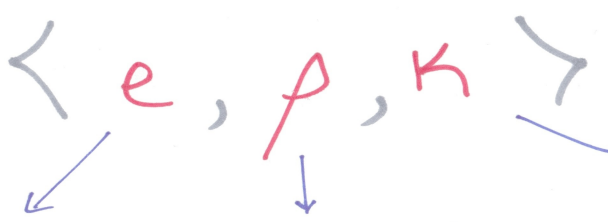
where $\kappa \approx E$ and ρ closes e.

$E[(d\ [])] \approx fn(d, \kappa)$

where $\kappa \approx E$.

States

$$s ::= \langle e, \rho, K \rangle \mid \langle d, K \rangle$$



$$\langle x, \rho, K \rangle \mapsto \langle d, K \rangle \text{ where } d = \rho(x)$$

$$\langle \lambda x.e, \rho, K \rangle \mapsto \langle\langle \lambda x.e, \rho \rangle, K \rangle$$

$$\langle e_0\, e_1, \rho, K \rangle \mapsto \langle e_0, \rho, ar(e_1, \rho, K) \rangle$$

$$\langle d, ar(e, \rho, K) \rangle \mapsto \langle e, fn(d, K) \rangle$$

$$\langle d', fn(d, K) \rangle \mapsto \langle e, \rho[x \mapsto d'], K \rangle \text{ where } d = \langle \lambda x.e, \rho \rangle$$

States

$s ::= \langle e, \rho, \sigma, \kappa \rangle \mid \langle d, \sigma, \kappa \rangle$

$\langle e, \rho, \sigma, \kappa \rangle$ CESK Machine

- Expression : Expr
- Environment : Var → Addr
- Store : Addr → Val
- Continuation : Kont

$$\langle x, \rho, \sigma, \kappa \rangle \mapsto \langle d, \sigma, \kappa \rangle \text{ where } d = \sigma(\rho(x))$$

$$\langle \lambda x.e, \rho, \sigma, \kappa \rangle \mapsto \langle\langle \lambda x.e, \rho \rangle, \sigma, \kappa \rangle$$

$$\langle e_0\, e_1, \rho, \sigma, \kappa \rangle \mapsto \langle e_0, \rho, \sigma, ar(e_1, \rho, \kappa) \rangle$$

$$\langle d, \sigma, ar(e, \rho, \kappa) \rangle \mapsto \langle e, \sigma, fn(d, \kappa) \rangle$$

$$\langle d', \sigma, fn(d, \kappa) \rangle \mapsto \langle e, \rho[x \mapsto a], \sigma[a \mapsto d'], \kappa \rangle$$

where $d = \langle \lambda x.e, \rho \rangle$

$a \notin dom(\sigma)$

States

$s ::= \langle e, \rho, \sigma, a \rangle \mid \langle d, \sigma, a \rangle$

$\langle e, \rho, \sigma, a \rangle$ CESK* Machine

- Expression : Expr
- Environment : Var → Addr
- Store : Addr → Val
- Continuation pointer : Addr

$$\langle x, \rho, \sigma, a \rangle \mapsto \langle d, \sigma, a \rangle \text{ where } d = \sigma(\rho(x))$$

$$\langle \lambda x.e, \rho, \sigma, a \rangle \mapsto \langle\langle \lambda x.e, \rho \rangle, \sigma, a \rangle$$

$$\langle e_0\, e_1, \rho, \sigma, a \rangle \mapsto \langle e_0, \rho, \sigma', a' \rangle \quad \sigma' = \sigma[a' \mapsto ar(e_1, \rho, a)] \quad a' \notin dom(\sigma)$$

$$\langle d, \sigma, a \rangle \mapsto \langle e, \rho, \sigma', a'' \rangle \quad \sigma' = \sigma[a'' \mapsto fn(d, a)] \quad a'' \notin dom(\sigma)$$

where $\sigma(a) = ar(e, \rho, a')$

$$\mapsto \langle e, \rho[x \mapsto a], \sigma[a \mapsto d'], a \rangle \text{ where } d = \langle \lambda x.e, \rho \rangle \quad a \notin dom(\sigma)$$

$\langle d', \sigma, a \rangle$

where $\sigma(a) = fn(d, a')$

States

$s ::= \langle e, \rho, \sigma, a \rangle \mid \langle d, \sigma, a \rangle$

$\langle e, \rho, \sigma, a \rangle$ CESK* Machine

Expression : Expr

Environment : Var → Addr

Store : Addr → Val

Continuation pointer : Addr

$\langle x, \rho, \sigma, a \rangle \mapsto \langle d, \sigma, a \rangle$ where $d = \sigma(\rho(x))$

$\langle \lambda x.e, \rho, \sigma, a \rangle \mapsto \langle\langle \lambda x.e, \rho \rangle, \sigma, a \rangle$

$\langle e_0\, e_1, \rho, \sigma, a \rangle \mapsto \langle e_0, \rho, \sigma', a' \rangle$

$\sigma' = \sigma[a' \mapsto ar(e_1, \rho, a)] \quad a' = alloc(s)$

$\langle d, \sigma, a \rangle \mapsto \langle e, \rho, \sigma', a'' \rangle$

where $\sigma(a) = ar(e, \rho, a')$

$\sigma' = \sigma[a'' \mapsto fn(d, a)] \quad a'' = alloc(s)$

$\mapsto \langle e, \rho[x \mapsto a], \sigma[a \mapsto d'], a \rangle$

where $d = \langle \lambda x.e, \rho \rangle$

$a = alloc(s)$

$\langle d', \sigma, a \rangle$

where $\sigma(a) = fn(d, a')$

States

$\hat{s} ::= \langle e, \rho, \sigma, a \rangle \mid \langle d, \sigma, a \rangle$

$\langle e, \rho, \sigma, a \rangle$ — CESK* Machine

- Expression : Expr
- Environment : Var → Addr
- Store : Addr → Val
- Continuation pointer : Addr

$$\langle x, \rho, \sigma, a \rangle \mapsto \langle d, \sigma, a \rangle \text{ where } d \ni \sigma(\rho(x))$$

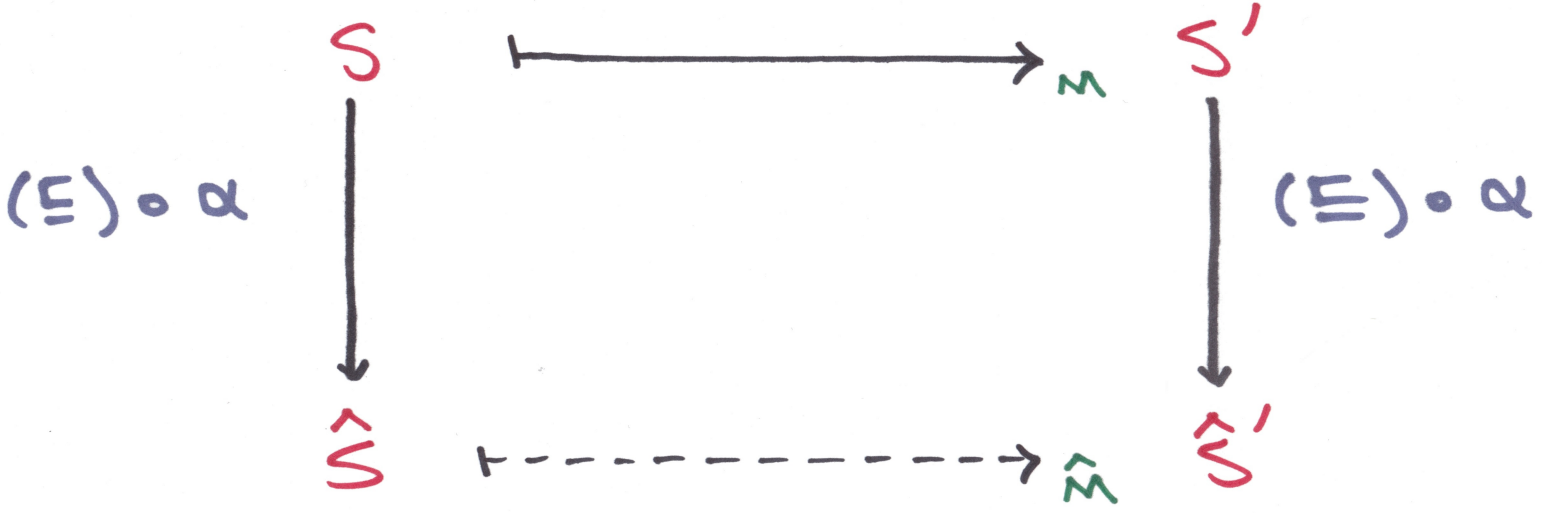
$$\langle \lambda x.e, \rho, \sigma, a \rangle \mapsto \langle\langle \lambda x.e, \rho \rangle, \sigma, a \rangle$$

$$\langle e_0\, e_1, \rho, \sigma, a \rangle \mapsto \langle e_0, \rho, \sigma', a' \rangle$$
$$\sigma' = \sigma \sqcup [a' \mapsto ar(e_1, \rho, a)] \quad a' = \widehat{alloc}(s)$$

$$\langle d, \sigma, a \rangle \mapsto \langle e, \rho, \sigma', a'' \rangle$$
where $\sigma(a) \ni ar(e, \rho, a')$
$$\sigma' = \sigma \sqcup [a'' \mapsto fn(d, a)] \quad a'' = \widehat{alloc}(s)$$

$$\langle d', \sigma, a \rangle \mapsto \langle e, \rho[x \mapsto a], \sigma \sqcup [a \mapsto d'], a \rangle$$
where $\sigma(a) \ni fn(d, a')$
where $d = \langle \lambda x.e, \rho \rangle$
$$a = \widehat{alloc}(s)$$

Soundness



Preprint Redex models

Abstracting Abstract Machines

http://lambda-calcul.us/aam/preprint.pdf

http://lambda-calcul.us/aam/redex/

THE END

Thank You