Abstracting abstract machines

David Van Horn and Matthew Might

Abstracting Abstract Machines

David Van Horn
Northeastern

Matthew Might
Utah

Challenges:

static analysis for reasoning about

- space consumption with laziness
- state + control with effects
- security with stack inspection
- blame with behavioral contracts
- safe parallelism with futures

Exp $\quad E ::= x \mid \lambda x.E \mid E\,E$

Val $\quad V ::= \lambda x.E$

$$(\lambda x.E)\,V \;\beta_v\; [V/x]E$$

$$\mathcal{E} ::= [\,] \mid (\mathcal{E}\,E) \mid (V\,\mathcal{E})$$

$\mathrm{eval}(E) = V$ if $E \longmapsto^* V$, where

$$\mathcal{E}[E_0] \longmapsto \mathcal{E}[E_1] \text{ if } E_0 \;\beta_v\; E_1$$

Env $\rho : Var \to (Val, Env)$

Cont $\kappa ::= mt \mid ar(E, \rho, \kappa) \mid fn(V, \rho, \kappa)$

$$[] \approx mt$$

$$\mathcal{E}[([]\ E)] \approx ar(E', \rho, \kappa),$$
where $(E', \rho) \approx E$, $\kappa \approx \mathcal{E}$

$$\mathcal{E}[(V\ [])] \approx fn(V', \rho, \kappa),$$
where $(V', \rho) \approx V$, $\kappa \approx \mathcal{E}$

$$\langle x, \rho, \kappa \rangle \longmapsto \langle V, \rho', \kappa \rangle \text{ if } \rho(x) = (V, \rho')$$

$$\langle E_0 E_1, \rho, \kappa \rangle \longmapsto \langle E_0, \rho, ar(E_1, \rho', \kappa) \rangle$$

$$\langle V, \rho, ar(E, \rho', \kappa) \rangle \longmapsto \langle E, \rho', fn(V, \rho', \kappa) \rangle$$

$$\langle V, \rho, fn(\lambda x.E, \rho', \kappa) \rangle \longmapsto \langle E, \rho'[x \mapsto (V, \rho)], \kappa \rangle$$

$$eval(E) = V \text{ if } \langle E, \emptyset, mt \rangle \mapsto^* \langle V', \rho, \kappa \rangle$$

where $(V', \rho) \approx V$

$$\{\langle E', \rho, \kappa\rangle \mid \langle E, \emptyset, \mathrm{mt}\rangle \mapsto^{*} \langle E', \rho, \kappa\rangle\}$$

Problem: produce a sound computable approximation

Answer: store allocate bindings & continuations in a bounded store

Step ①: store allocate bindings

Store $\sigma : Addr \to Sto$

$\langle E, \rho, \sigma, \kappa \rangle$

Env $\rho : Var \to Addr$

$\langle x, \rho, \sigma, \kappa \rangle \longmapsto \langle V, \rho', \sigma, \kappa \rangle$ if $\sigma(\rho(x)) = (V, \rho')$

$\langle E_0 E_1, \rho, \sigma, \kappa \rangle \longmapsto \langle E_0, \rho, \sigma, ar(E_1, \rho', \kappa) \rangle$

$\langle V, \rho, \sigma, ar(E, \rho', \kappa) \rangle \longmapsto \langle E, \rho', \sigma, fn(V, \rho', \kappa) \rangle$

$\langle V, \rho, \sigma, fn(\lambda x.E, \rho', \kappa) \rangle \longmapsto \langle E, \rho'[x \mapsto b], \sigma[b \mapsto (V, \rho)], \kappa \rangle$

where $b = alloc(s)$

Step ②: store allocate continuations

Cont $\quad \kappa ::= mt \mid ar(E, \rho, \kappa) \mid fn(V, \rho, \kappa)$

Cont $\quad \kappa ::= mt \mid ar(E, \rho, a) \mid fn(V, \rho, a)$

$$\langle E, \rho, \sigma, a \rangle$$

$$\langle x, \rho, \sigma, a \rangle \mapsto \langle V, \rho', \sigma, a \rangle \text{ if } \sigma(\rho(x)) = (V, \rho')$$

$$\langle E_0 E_1, \rho, \sigma, a \rangle \mapsto \langle E_0, \rho, \sigma\,[b \mapsto ar(E_1, \rho, a)], b \rangle$$

$$\langle V, \rho, \sigma, a \rangle \mapsto \langle E, \rho', \sigma\,[b \mapsto fn(V, \rho', a')], b \rangle$$
$$\text{if } ar(E, \rho', a') = d(a)$$

$$\langle V, \rho, \sigma, a \rangle \mapsto \langle E, \rho'[x \mapsto b], \sigma\,[b \mapsto (V, \rho)], a' \rangle$$
$$\text{if } ar(E, \rho', a') = d(a)$$

where $b = alloc(s)$

Store $\sigma : \mathrm{Addr} \to \mathcal{P}(\mathrm{Sto})$

$$\langle x, \rho, \sigma, a \rangle \mapsto \langle V, \rho', \sigma, a \rangle \text{ if } \sigma(\rho(x)) \ni (V, \rho')$$

$$\langle E_0 E_1, \rho, \sigma, a \rangle \mapsto \langle E_0, \rho, \sigma\,[b \mapsto ar(E_1, \rho', a)], b \rangle$$

$$\langle V, \rho, \sigma, a \rangle \mapsto \langle E, \rho', \sigma\,[b \mapsto fn(V, \rho', a')], b \rangle$$
$$\text{if } ar(E, \rho', a') \ni d(a)$$

$$\langle V, \rho, \sigma, a \rangle \mapsto \langle E, \rho'[x \mapsto b], \sigma\,[b \mapsto (V, \rho)], a' \rangle$$
$$\text{if } ar(E, \rho', a') \ni d(a)$$

where $b = alloc(s)$

Step ③: Bound the store

Finally, we abstract:

$$\mathtt{alloc} : \mathit{State} \to \mathit{Addr}$$

$$\widehat{\mathtt{alloc}} : \mathit{State} \to \widehat{\mathit{Addr}}$$

$$\widehat{\mathit{Addr}} \subseteq_{fin} \mathit{Addr}$$

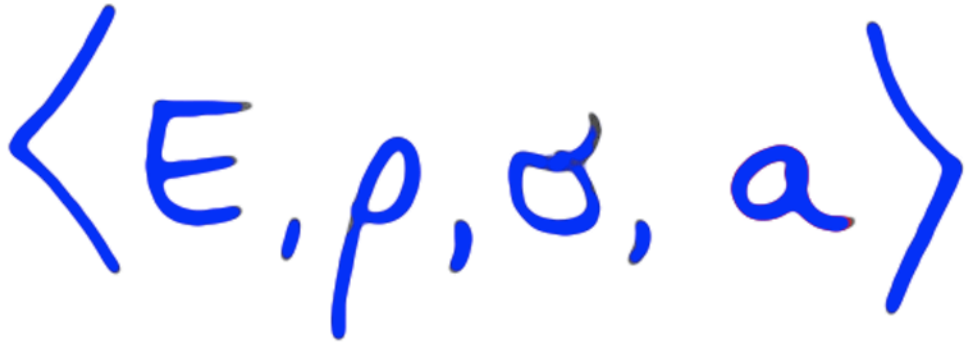
$$\langle x, \rho, \sigma, a \rangle \mapsto \langle V, \rho', \sigma, a \rangle \text{ if } \sigma(\rho(x)) \ni (V, \rho')$$

$$\langle E_0 E_1, \rho, \sigma, a \rangle \mapsto \langle E_0, \rho, \sigma \sqcup [b \mapsto ar(E_1, \rho', a)], b \rangle$$

$$\langle V, \rho, \sigma, a \rangle \mapsto \langle E, \rho', \sigma \sqcup [b \mapsto fn(V, \rho', a')], b \rangle$$
$$\text{if } ar(E, \rho', a') \ni \sigma(a)$$

$$\langle V, \rho, \sigma, a \rangle \mapsto \langle E, \rho'[x \mapsto b], \sigma \sqcup [b \mapsto (V, \rho)], a' \rangle$$
$$\text{if } ar(E, \rho', a') \ni \sigma(a)$$

where $b = \widehat{alloc}(s)$


⟨E, ρ, σ, a⟩

Challenges:

static analysis for reasoning about

- space consumption with laziness
- state + control with effects
- security with stack inspection
- blame with behavioral contracts
- safe parallelism with futures

Challenges:

static analysis for reasoning about

- space consi[illegible] with laziness
- state + co[illegible]h effects
- security with[illegible] inspection
- blame with behavioral contracts
- safe parallelism with futures

THE END


Sharpie
24
Marks on most surfaces
Marque sur la plupart des surfaces
Escribe en la mayoria de las superficies
PERMANENT MARKER
MARQUEUR PERMANENT
MARCADOR PERMANENTE
FINE
FINA
WRITE OUT LOUD!