A Spectrum of Type Soundness and Performance

By B. Greenman & M. Felleisen
Presented by Artem Pelenitsyn

Feb 25, 2019

THREE FLAVORS OF MIGRATORY TYPING

- Higher-Order Contracts
- Constructor-checking
- Erasing

Safety

Performance

Questions:

- How do the **logical implications** of the three approaches compare?
- How do the three approaches compare with respect to **performance**?

0. COMMON LANGUAGE

Syntax

$$
\begin{array}{lcl}
e_{\mathsf{D}} & = & x \mid \lambda x.\, e_{\mathsf{D}} \mid i \mid \langle e_{\mathsf{D}}, e_{\mathsf{D}} \rangle \mid e_{\mathsf{D}}\ e_{\mathsf{D}} \mid \boxed{op^1\ e_{\mathsf{D}} \mid op^2\ e_{\mathsf{D}}\ e_{\mathsf{D}}} \mid \boxed{\mathsf{stat}\ \tau\ e_{\mathsf{S}}} \\
e_{\mathsf{S}} & = & x \mid \lambda(x:\tau).\, e_{\mathsf{S}} \mid i \mid \langle e_{\mathsf{S}}, e_{\mathsf{S}} \rangle \mid e_{\mathsf{S}}\ e_{\mathsf{S}} \mid \boxed{op^1\ e_{\mathsf{S}} \mid op^2\ e_{\mathsf{S}}\ e_{\mathsf{S}}} \mid \boxed{\mathsf{dyn}\ \tau\ e_{\mathsf{D}}} \\
\tau & = & \mathsf{Int} \mid \mathsf{Nat} \mid \tau \times \tau \mid \tau \Rightarrow \tau \\
i & \in & \mathbb{Z} \\
op^1 & = & \mathsf{fst} \mid \mathsf{snd} \\
op^2 & = & \mathsf{sum} \mid \mathsf{quotient}
\end{array}
$$

$$\boxed{\Delta : op^1 \times \tau \longrightarrow \tau}$$

$$
\begin{array}{lcl}
\Delta(\mathsf{fst}, \tau_0 \times \tau_1) & = & \tau_0 \\
\Delta(\mathsf{snd}, \tau_0 \times \tau_1) & = & \tau_1
\end{array}
$$

$$\boxed{\Delta : op^2 \times \tau \times \tau \longrightarrow \tau}$$

$$
\begin{array}{lcl}
\Delta(op^2, \mathsf{Nat}, \mathsf{Nat}) & = & \mathsf{Nat} \\
\Delta(op^2, \mathsf{Int}, \mathsf{Int}) & = & \mathsf{Int}
\end{array}
$$

$$\boxed{\tau \leqslant: \tau}$$

$$\frac{}{\mathsf{Nat} \leqslant: \mathsf{Int}}$$

$$\frac{\tau_d' \leqslant: \tau_d \quad \tau_c \leqslant: \tau_c'}{\tau_d \Rightarrow \tau_c \leqslant: \tau_d' \Rightarrow \tau_c'} \qquad \frac{\tau_0 \leqslant: \tau_0' \quad \tau_1 \leqslant: \tau_1'}{\tau_0 \times \tau_1 \leqslant: \tau_0' \times \tau_1'}$$

$$\frac{}{\tau \leqslant: \tau} \qquad \frac{\tau \leqslant: \tau' \quad \tau' \leqslant: \tau''}{\tau \leqslant: \tau''}$$

Typing

$\boxed{\Gamma \vdash e}$

$$\frac{x \in \Gamma}{\Gamma \vdash x} \quad \frac{x, \Gamma \vdash e}{\Gamma \vdash \lambda x.\, e} \quad \frac{}{\Gamma \vdash i} \quad \frac{\Gamma \vdash e_0 \quad \Gamma \vdash e_1}{\Gamma \vdash \langle e_0, e_1 \rangle} \quad \frac{\Gamma \vdash e_0 \quad \Gamma \vdash e_1}{\Gamma \vdash e_0\ e_1} \quad \frac{\Gamma \vdash e}{\Gamma \vdash op^1\ e} \quad \frac{\Gamma \vdash e_0 \quad \Gamma \vdash e_1}{\Gamma \vdash op^2\ e_0\ e_1}$$

$$\frac{}{\Gamma \vdash \mathsf{Err}} \quad \boxed{\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathsf{stat}\ \tau\ e}}$$

$\boxed{\Gamma \vdash e : \tau}$

$$\frac{(x{:}\tau) \in \Gamma}{\Gamma \vdash x : \tau} \quad \frac{(x{:}\tau_d), \Gamma \vdash e : \tau_c}{\Gamma \vdash \lambda(x{:}\tau_d).\, e : \tau_d \Rightarrow \tau_c} \quad \frac{i \in \mathbb{N}}{\Gamma \vdash i : \mathsf{Nat}} \quad \frac{}{\Gamma \vdash i : \mathsf{Int}} \quad \frac{\Gamma \vdash e_0 : \tau_0 \quad \Gamma \vdash e_1 : \tau_1}{\Gamma \vdash \langle e_0, e_1 \rangle : \tau_0 \times \tau_1}$$

$$\frac{\Gamma \vdash e_0 : \tau_d \Rightarrow \tau_c \quad \Gamma \vdash e_1 : \tau_d}{\Gamma \vdash e_0\ e_1 : \tau_c} \quad \frac{\begin{array}{c}\Gamma \vdash e_0 : \tau_0 \\ \Delta(op^1, \tau_0) = \tau\end{array}}{\Gamma \vdash op^1\ e_0 : \tau} \quad \frac{\begin{array}{c}\Gamma \vdash e_0 : \tau_0 \quad \Gamma \vdash e_1 : \tau_1 \\ \Delta(op^2, \tau_0, \tau_1) = \tau\end{array}}{\Gamma \vdash op^2\ e_0\ e_1 : \tau} \quad \frac{\begin{array}{c}\Gamma \vdash e : \tau' \\ \tau' \leqslant: \tau\end{array}}{\Gamma \vdash e : \tau}$$

$$\frac{}{\Gamma \vdash \mathsf{Err} : \tau} \quad \boxed{\frac{\Gamma \vdash e}{\Gamma \vdash \mathsf{dyn}\ \tau\ e : \tau}}$$

$\boxed{\text{Typing Syntax}}$ extends Surface Syntax

$$\begin{array}{lcl} e & = & e_S \mid e_D \mid \mathsf{Err} \\ \mathsf{Err} & = & \mathsf{BndryErr} \mid \mathsf{TagErr} \\ \Gamma & = & \cdot \mid x, \Gamma \mid (x{:}\tau), \Gamma \end{array}$$

Common Semantics: Syntax & PrimOps

$\boxed{\text{Evaluation Syntax}}$ extends Surface Syntax

$$
\begin{array}{lcl}
e & = & x \mid v \mid \langle e, e\rangle \mid e\ e \mid op^1\ e \mid op^2\ e\ e \mid \mathsf{dyn}\ \tau\ e \mid \mathsf{stat}\ \tau\ e \mid \mathsf{Err} \\
v & = & i \mid \langle v, v\rangle \mid \lambda x.\, e \mid \lambda(x : \tau).\, e \\
\mathsf{Err} & = & \mathsf{BndryErr} \mid \mathsf{TagErr} \\
r & = & v \mid \mathsf{Err} \\
E^{\bullet} & = & [\,] \mid E^{\bullet}\ e \mid v\ E^{\bullet} \mid \langle E^{\bullet}, e\rangle \mid \langle v, E^{\bullet}\rangle \mid op^1\ E^{\bullet} \mid op^2\ E^{\bullet}\ e \mid op^2\ v\ E^{\bullet} \\
E & = & E^{\bullet} \mid E\ e \mid v\ E \mid \langle E, e\rangle \mid \langle v, E\rangle \mid op^1\ E \mid op^2\ E\ e \mid op^2\ v\ E \mid \mathsf{dyn}\ \tau\ E \mid \mathsf{stat}\ \tau\ E
\end{array}
$$

$\boxed{\delta : op^1 \times v \longrightarrow v}$

$$
\begin{array}{ll}
\delta(\mathsf{fst}, \langle v_0, v_1\rangle) & = v_0 \\
\delta(\mathsf{snd}, \langle v_0, v_1\rangle) & = v_1
\end{array}
$$

$\boxed{\delta : op^2 \times v \times v \longrightarrow r}$

$$
\begin{array}{ll}
\delta(\mathsf{sum}, i_0, i_1) & = i_0 + i_1 \\
\delta(\mathsf{quotient}, i_0, 0) & = \mathsf{BndryErr} \\
\delta(\mathsf{quotient}, i_0, i_1) & = \lfloor i_0 / i_1 \rfloor \\
\quad \text{if } i_1 \neq 0 &
\end{array}
$$

Common Semantics: Reductions

$\boxed{e \rhd_S e}$

$(\lambda(x:\tau).\,e)\;v \;\rhd_S\; e[x\leftarrow v]$

$op^1\;v \;\rhd_S\; \delta(op^1, v)$

$op^2\;v_0\;v_1 \;\rhd_S\; \delta(op^2, v_0, v_1)$

$\boxed{e \rhd_D e}$

$v_0\;v_1 \;\rhd_D\; \mathsf{TagErr}$ if $v_0 \in \mathbb{Z}$ or $v_0 = \langle v, v' \rangle$

$(\lambda x.\,e)\;v \;\rhd_D\; e[x\leftarrow v]$

$op^1\;v \;\rhd_D\; \mathsf{TagErr}$ if $\delta(op^1, v)$ is undefined

$op^1\;v \;\rhd_D\; \delta(op^1, v)$

$op^2\;v_0\;v_1 \;\rhd_D\; \mathsf{TagErr}$ if $\delta(op^2, v_0, v_1)$ is undefined

$op^2\;v_0\;v_1 \;\rhd_D\; \delta(op^2, v_0, v_1)$

1. HIGHER-ORDER EMBEDDING

Higher-Order Embedding: Monitor Syntax, D_H and S_H for boundary translation

$\boxed{\text{Language H}}$ extends Evaluation Syntax

$$v \;=\; \ldots \mid \boxed{\mathsf{mon}\,(\tau \Rightarrow \tau)\, v}$$

$\boxed{\mathcal{D}_{\mathsf{H}} : \tau \times v \longrightarrow e}$

$$\begin{aligned}
&\mathcal{D}_{\mathsf{H}}(\tau_d \Rightarrow \tau_c, v) && = \mathsf{mon}\,(\tau_d \Rightarrow \tau_c)\, v \\
&\quad \text{if } v = \lambda x.\, e \text{ or } v = \mathsf{mon}\,\tau'\, v' && \\
&\mathcal{D}_{\mathsf{H}}(\tau_0 \times \tau_1, \langle v_0, v_1 \rangle) && = \langle \mathsf{dyn}\ \tau_0\ v_0, \mathsf{dyn}\ \tau_1\ v_1 \rangle \\
&\mathcal{D}_{\mathsf{H}}(\mathsf{Int}, i) && = i \\
&\mathcal{D}_{\mathsf{H}}(\mathsf{Nat}, i) && = i \\
&\quad \text{if } i \in \mathbb{N} && \\
&\mathcal{D}_{\mathsf{H}}(\tau, v) && = \mathsf{BndryErr} \\
&\quad \text{otherwise} &&
\end{aligned}$$

$\boxed{\mathcal{S}_{\mathsf{H}} : \tau \times v \longrightarrow e}$

$$\begin{aligned}
&\mathcal{S}_{\mathsf{H}}(\tau_d \Rightarrow \tau_c, v) && = \mathsf{mon}\,(\tau_d \Rightarrow \tau_c)\, v \\
&\mathcal{S}_{\mathsf{H}}(\tau_0 \times \tau_1, \langle v_0, v_1 \rangle) && = \langle \mathsf{stat}\ \tau_0\ v_0, \mathsf{stat}\ \tau_1\ v_1 \rangle \\
&\mathcal{S}_{\mathsf{H}}(\mathsf{Int}, v) && = v \\
&\mathcal{S}_{\mathsf{H}}(\mathsf{Nat}, v) && = v
\end{aligned}$$

Higher-Order Embedding: reductions & evaluation

$\boxed{e \rhd_{\text{H-S}} e}$ extends $\rhd_{\text{S}}$

$$\begin{array}{lll} \text{dyn}\ \tau\ v & \rhd_{\text{H-S}} & \mathcal{D}_{\text{H}}(\tau, v) \\ (\text{mon}\,(\tau_d \Rightarrow \tau_c)\,v_f)\ v & \rhd_{\text{H-S}} & \text{dyn}\ \tau_c\ (v_f\ e') \\ \quad \text{where } e' = \text{stat}\ \tau_d\ v & & \end{array}$$

$\boxed{e \rhd_{\text{H-D}} e}$ extends $\rhd_{\text{D}}$

$$\begin{array}{lll} \text{stat}\ \tau\ v & \rhd_{\text{H-D}} & \mathcal{S}_{\text{H}}(\tau, v) \\ (\text{mon}\,(\tau_d \Rightarrow \tau_c)\,v_f)\ v & \rhd_{\text{H-D}} & \text{stat}\ \tau_c\ (v_f\ e') \\ \quad \text{where } e' = \text{dyn}\ \tau_d\ v & & \end{array}$$

$\boxed{e \rightarrow^*_{\text{H-S}} e}$ reflexive, transitive closure of $\rightarrow_{\text{H-S}}$

$$\begin{array}{lll} E^\bullet[e] & \rightarrow_{\text{H-S}} & E^\bullet[e'] \\ \quad \text{if } e \rhd_{\text{H-S}} e' & & \\ E[\text{stat}\ \tau\ E^\bullet[e]] & \rightarrow_{\text{H-S}} & E[\text{stat}\ \tau\ E^\bullet[e']] \\ \quad \text{if } e \rhd_{\text{H-S}} e' & & \\ E[\text{dyn}\ \tau\ E^\bullet[e]] & \rightarrow_{\text{H-S}} & E[\text{dyn}\ \tau\ E^\bullet[e']] \\ \quad \text{if } e \rhd_{\text{H-D}} e' & & \\ E[\text{Err}] & \rightarrow_{\text{H-S}} & \text{Err} \end{array}$$

$\boxed{e \rightarrow^*_{\text{H-D}} e}$ reflexive, transitive closure of $\rightarrow_{\text{H-D}}$

$$\begin{array}{lll} E^\bullet[e] & \rightarrow_{\text{H-D}} & E^\bullet[e'] \\ \quad \text{if } e \rhd_{\text{H-D}} e' & & \\ E[\text{stat}\ \tau\ E^\bullet[e]] & \rightarrow_{\text{H-D}} & E[\text{stat}\ \tau\ E^\bullet[e']] \\ \quad \text{if } e \rhd_{\text{H-S}} e' & & \\ E[\text{dyn}\ \tau\ E^\bullet[e]] & \rightarrow_{\text{H-D}} & E[\text{dyn}\ \tau\ E^\bullet[e']] \\ \quad \text{if } e \rhd_{\text{H-D}} e' & & \\ E[\text{Err}] & \rightarrow_{\text{H-D}} & \text{Err} \end{array}$$

Higher-Order Embedding: Typing & Soundness

$\boxed{\Gamma \vdash_{\mathsf{H}} e}$ extends $\Gamma \vdash e$

$$\frac{\Gamma \vdash_{\mathsf{H}} v : \tau_d \Rightarrow \tau_c}{\Gamma \vdash_{\mathsf{H}} \mathsf{mon}\,(\tau_d \Rightarrow \tau_c)\, v}$$

$\boxed{\Gamma \vdash_{\mathsf{H}} e : \tau}$ extends $\Gamma \vdash e : \tau$

$$\frac{\Gamma \vdash_{\mathsf{H}} v}{\Gamma \vdash_{\mathsf{H}} \mathsf{mon}\,(\tau_d \Rightarrow \tau_c)\, v : \tau_d \Rightarrow \tau_c}$$

Theorem 2.1 : *static* H-*soundness*

If $e \in e_{\mathsf{S}}$ and $\vdash e : \tau$

then $\vdash_{\mathsf{H}} e : \tau$ and one of the following holds:

- $e \rightarrow^*_{\mathsf{H\text{-}S}} v$ and $\vdash_{\mathsf{H}} v : \tau$
- $e \rightarrow^*_{\mathsf{H\text{-}S}} E[\mathsf{dyn}\ \tau'\ E^{\bullet}[e']]$ and $e' \rhd_{\mathsf{H\text{-}D}} \mathsf{TagErr}$
- $e \rightarrow^*_{\mathsf{H\text{-}S}} \mathsf{BndryErr}$
- e diverges

Theorem 2.2 : *dynamic* H-*soundness*

If $e \in e_{\mathsf{D}}$ and $\vdash e$

then $\vdash_{\mathsf{H}} e$ and one of the following holds:

- $e \rightarrow^*_{\mathsf{H\text{-}D}} v$ and $\vdash_{\mathsf{H}} v$
- $e \rightarrow^*_{\mathsf{H\text{-}D}} E[e']$ and $e' \rhd_{\mathsf{H\text{-}D}} \mathsf{TagErr}$
- $e \rightarrow^*_{\mathsf{H\text{-}D}} \mathsf{BndryErr}$
- e diverges

2. ERASURE EMBEDDING

Erasure Embedding: No Syntax Change, Trivial Semantics

$\boxed{\mathcal{D}_{\mathsf{E}} : \tau \times v \longrightarrow e}$

$\mathcal{D}_{\mathsf{E}}(\tau, v) = v$

$\boxed{e \rhd_{\mathsf{E\text{-}S}} e}$ extends $\rhd_{\mathsf{S}}$

$$
\begin{array}{lll}
\mathsf{dyn}\ \tau\ v & \rhd_{\mathsf{E\text{-}S}} & \mathcal{D}_{\mathsf{E}}(\tau, v) \\
\mathsf{stat}\ \tau\ v & \rhd_{\mathsf{E\text{-}S}} & \mathcal{S}_{\mathsf{E}}(\tau, v) \\
(\lambda x.\, e)\ v & \rhd_{\mathsf{E\text{-}S}} & e[x \leftarrow v] \\
v_0\ v_1 & \rhd_{\mathsf{E\text{-}S}} & \mathsf{TagErr} \\
\quad \text{if } v_0 \in \mathbb{Z} \text{ or } v_0 = \langle v, v' \rangle & & \\
op^1\ v & \rhd_{\mathsf{E\text{-}S}} & \mathsf{TagErr} \\
\quad \text{if } \delta(op^1, v) \text{ is undefined} & & \\
op^2\ v_0\ v_1 & \rhd_{\mathsf{E\text{-}S}} & \mathsf{TagErr} \\
\quad \text{if } \delta(op^2, v_0, v_1) \text{ is undefined} & &
\end{array}
$$

$\boxed{e \rightarrow^*_{\mathsf{E\text{-}S}} e}$ reflexive, transitive closure of $\rightarrow_{\mathsf{E\text{-}S}}$

$$
\begin{array}{lll}
E[e] & \rightarrow_{\mathsf{E\text{-}S}} & E[e'] \\
\quad \text{if } e \rhd_{\mathsf{E\text{-}S}} e' & & \\
E[\mathsf{Err}] & \rightarrow_{\mathsf{E\text{-}S}} & \mathsf{Err}
\end{array}
$$

$\boxed{\mathcal{S}_{\mathsf{E}} : \tau \times v \longrightarrow e}$

$\mathcal{S}_{\mathsf{E}}(\tau, v) = v$

$\boxed{e \rhd_{\mathsf{E\text{-}D}} e}$ extends $\rhd_{\mathsf{D}}$

$$
\begin{array}{lll}
\mathsf{stat}\ \tau\ v & \rhd_{\mathsf{E\text{-}D}} & \mathcal{S}_{\mathsf{E}}(\tau, v) \\
\mathsf{dyn}\ \tau\ v & \rhd_{\mathsf{E\text{-}D}} & \mathcal{D}_{\mathsf{E}}(\tau, v) \\
(\lambda(x : \tau_d).\, e)\ v & \rhd_{\mathsf{E\text{-}D}} & e[x \leftarrow v]
\end{array}
$$

$\boxed{e \rightarrow^*_{\mathsf{E\text{-}D}} e}$ reflexive, transitive closure of $\rightarrow_{\mathsf{E\text{-}D}}$

$$
\begin{array}{lll}
E[e] & \rightarrow_{\mathsf{E\text{-}D}} & E[e'] \\
\quad \text{if } e \rhd_{\mathsf{E\text{-}D}} e' & & \\
E[\mathsf{Err}] & \rightarrow_{\mathsf{E\text{-}D}} & \mathsf{Err}
\end{array}
$$

Erasure Embedding: Typing & Soundness

$\boxed{\Gamma \vdash_{\mathsf{E}} e}$ (selected rules)

$$\frac{x, \Gamma \vdash_{\mathsf{E}} e}{\Gamma \vdash_{\mathsf{E}} \lambda x.\, e} \qquad \frac{(x:\tau), \Gamma \vdash_{\mathsf{E}} e}{\Gamma \vdash_{\mathsf{E}} \lambda(x:\tau).\, e} \qquad \frac{\Gamma \vdash_{\mathsf{E}} e}{\Gamma \vdash_{\mathsf{E}} \mathsf{dyn}\ \tau\ e} \qquad \frac{\Gamma \vdash_{\mathsf{E}} e}{\Gamma \vdash_{\mathsf{E}} \mathsf{stat}\ \tau\ e}$$

Theorem 2.4 : *static* E-*soundness*

If $e \in e_{\mathsf{S}}$ and $\vdash e : \tau$
then $\vdash_{\mathsf{E}} e$ and one of the following holds:

- $e \rightarrow^{*}_{\mathsf{E\text{-}S}} v$ and $\vdash_{\mathsf{E}} v$
- $e \rightarrow^{*}_{\mathsf{E\text{-}S}} \mathsf{TagErr}$
- $e \rightarrow^{*}_{\mathsf{E\text{-}S}} \mathsf{BndryErr}$
- e diverges

Theorem 2.5 : *dynamic* E-*soundness*

If $e \in e_{\mathsf{D}}$ and $\vdash e$
then $\vdash_{\mathsf{E}} e$ and one of the following holds:

- $e \rightarrow^{*}_{\mathsf{E\text{-}D}} v$ and $\vdash_{\mathsf{E}} v$
- $e \rightarrow^{*}_{\mathsf{E\text{-}D}} \mathsf{TagErr}$
- $e \rightarrow^{*}_{\mathsf{E\text{-}D}} \mathsf{BndryErr}$
- e diverges

Erasure: Bound-Free Programs Don't Rise TagErr

Theorem 2.6 : *boundary-free* E-*soundness*

If $e \in e_{\mathsf{S}}$ and $\vdash e : \tau$ and e does not contain a subexpression $(\mathsf{dyn}\ \tau'\ e')$ then one of the following holds:

- $e \rightarrow^{*}_{\mathsf{E\text{-}S}} v$ and $\vdash v : \tau$
- $e \rightarrow^{*}_{\mathsf{E\text{-}S}} \mathsf{BndryErr}$
- e diverges

3. FIRST-ORDER EMBEDDING

First-Order Embedding: Motivations & Ideas

1. Erasure's logical guarantees are too weak.
2. Higher-order's run-time monitoring is impractical from two PoWs:
   a. Implementation
   b. Usage
3. Middle ground in needed.

Idea: Purpose of types — prevent evaluation from applying a typed **elimination form** to a value outside its domain.

Mechanism: a type-directed rewriting pass **over typed code** to defend against untyped inputs.

Defence: type-constructor checks.

First-Order Embedding: Syntax & Translation

$\boxed{\text{Language 1}}$ extends Evaluation Syntax

$$
\begin{array}{lcl}
e & = & \ldots \mid \boxed{\mathsf{chk}\ K\ e} \mid \mathsf{dyn}\ e \mid \mathsf{stat}\ e \\
E^{\bullet} & = & \ldots \mid \mathsf{chk}\ K\ E^{\bullet} \\
E & = & \ldots \mid \mathsf{chk}\ K\ E \mid \mathsf{dyn}\ E \mid \mathsf{stat}\ E \\
K & = & \mathsf{Int} \mid \mathsf{Nat} \mid \mathsf{Pair} \mid \mathsf{Fun} \mid \mathsf{Any}
\end{array}
$$

$\boxed{\lfloor \cdot \rfloor : \tau \longrightarrow K}$

$$
\begin{array}{lcl}
\lfloor \mathsf{Int} \rfloor & = & \mathsf{Int} \\
\lfloor \mathsf{Nat} \rfloor & = & \mathsf{Nat} \\
\lfloor \tau_0 \times \tau_1 \rfloor & = & \mathsf{Pair} \\
\lfloor \tau_d \Rightarrow \tau_c \rfloor & = & \mathsf{Fun}
\end{array}
$$

First-Order Embedding: Reduction of Boundaries

$$\boxed{\mathcal{D}_1 : \tau \times v \longrightarrow v}$$

$$\mathcal{D}_1(\tau, v) = \mathcal{X}(\lfloor \tau \rfloor, v)$$

$$\boxed{\mathcal{S}_1 : \tau \times v \longrightarrow v}$$

$$\mathcal{S}_1(\tau, v) = v$$

$$\boxed{\mathcal{X} : K \times v \longrightarrow v}$$

$$\begin{aligned}
\mathcal{X}(\mathsf{Fun}, \lambda x.\, e) &= \lambda x.\, e \\
\mathcal{X}(\mathsf{Fun}, \lambda(x:\tau).\, e) &= \lambda(x:\tau).\, e \\
\mathcal{X}(\mathsf{Pair}, \langle v_0, v_1 \rangle) &= \langle v_0, v_1 \rangle
\end{aligned}$$

$$\begin{aligned}
\mathcal{X}(\mathsf{Int}, i) &= i \\
\mathcal{X}(\mathsf{Nat}, i) &= i \\
&\quad \text{if } i \in \mathbb{N} \\
\mathcal{X}(K, v) &= \mathsf{BndryErr} \\
&\quad \text{otherwise}
\end{aligned}$$

First-Order Embedding: Reductions & Evaluation

$\boxed{e \rhd_{\text{1-S}} e}$ extends and overrides $\rhd_{\text{S}}$

$$
\begin{array}{lll}
\mathsf{dyn}\ v & \rhd_{\text{1-S}} & v \\
\mathsf{dyn}\ \tau\ v & \rhd_{\text{1-S}} & \mathcal{D}(\tau, v) \\
\mathsf{chk}\ K\ v & \rhd_{\text{1-S}} & \mathcal{X}(K, v) \\
(\lambda(x:\tau).\ e)\ v & \rhd_{\text{1-S}} & \mathsf{BndryErr} \\
\quad \text{if } \mathcal{X}(\lfloor\tau\rfloor, v) = \mathsf{BndryErr} & & \\
(\lambda(x:\tau).\ e)\ v & \rhd_{\text{1-S}} & e[x \leftarrow \mathcal{X}(\lfloor\tau\rfloor, v)] \\
\quad \text{if } \mathcal{X}(\lfloor\tau\rfloor, v) \neq \mathsf{BndryErr} & & \\
(\lambda x.\ e)\ v & \rhd_{\text{1-S}} & \mathsf{dyn}\ (e[x \leftarrow v])
\end{array}
$$

$\boxed{e \rhd_{\text{1-D}} e}$ extends $\rhd_{\text{D}}$

$$
\begin{array}{lll}
\mathsf{stat}\ v & \rhd_{\text{1-D}} & v \\
\mathsf{stat}\ \tau\ v & \rhd_{\text{1-D}} & \mathcal{S}(\tau, v) \\
(\lambda(x:\tau).\ e)\ v & \rhd_{\text{1-D}} & \mathsf{BndryErr} \\
\quad \text{if } \mathcal{X}(\lfloor\tau\rfloor, v) = \mathsf{BndryErr} & & \\
(\lambda(x:\tau).\ e)\ v & \rhd_{\text{1-D}} & \mathsf{stat}\ (e[x \leftarrow \mathcal{X}(\lfloor\tau\rfloor, v)]) \\
\quad \text{if } \mathcal{X}(\lfloor\tau\rfloor, v) \neq \mathsf{BndryErr} & &
\end{array}
$$

$\boxed{e \rightarrow^{*}_{\text{1-S}} e}$ reflexive, transitive closure of $\rightarrow_{\text{1-S}}$

$$
\begin{array}{lll}
E^{\bullet}[e] & \rightarrow_{\text{1-S}} & E^{\bullet}[e'] \\
\quad \text{if } e \rhd_{\text{1-S}} e' & & \\
E[\mathsf{stat}\ \tau\ E^{\bullet}[e]] & \rightarrow_{\text{1-S}} & E[\mathsf{stat}\ \tau\ E^{\bullet}[e']] \\
\quad \text{if } e \rhd_{\text{1-S}} e' & & \\
E[\mathsf{dyn}\ \tau\ E^{\bullet}[e]] & \rightarrow_{\text{1-S}} & E[\mathsf{dyn}\ \tau\ E^{\bullet}[e']] \\
\quad \text{if } e \rhd_{\text{1-D}} e' & & \\
E[\mathsf{Err}] & \rightarrow_{\text{1-S}} & \mathsf{Err}
\end{array}
$$

$\boxed{e \rightarrow^{*}_{\text{1-D}} e}$ reflexive, transitive closure of $\rightarrow_{\text{1-D}}$

$$
\begin{array}{lll}
E^{\bullet}[e] & \rightarrow_{\text{1-D}} & E^{\bullet}[e'] \\
\quad \text{if } e \rhd_{\text{1-D}} e' & & \\
E[\mathsf{stat}\ \tau\ E^{\bullet}[e]] & \rightarrow_{\text{1-D}} & E[\mathsf{stat}\ \tau\ E^{\bullet}[e']] \\
\quad \text{if } e \rhd_{\text{1-S}} e' & & \\
E[\mathsf{dyn}\ \tau\ E^{\bullet}[e]] & \rightarrow_{\text{1-D}} & E[\mathsf{dyn}\ \tau\ E^{\bullet}[e']] \\
\quad \text{if } e \rhd_{\text{1-D}} e' & & \\
E[\mathsf{Err}] & \rightarrow_{\text{1-D}} & \mathsf{Err}
\end{array}
$$

First-Order Embedding: Soundness

Theorem 2.7 : *static* 1-*soundness*

If $e \in e_S$ and $\vdash e : \tau$
then $\vdash e : \tau \rightsquigarrow e''$ and $\vdash_1 e'' : \lfloor \tau \rfloor$
and one of the following holds:

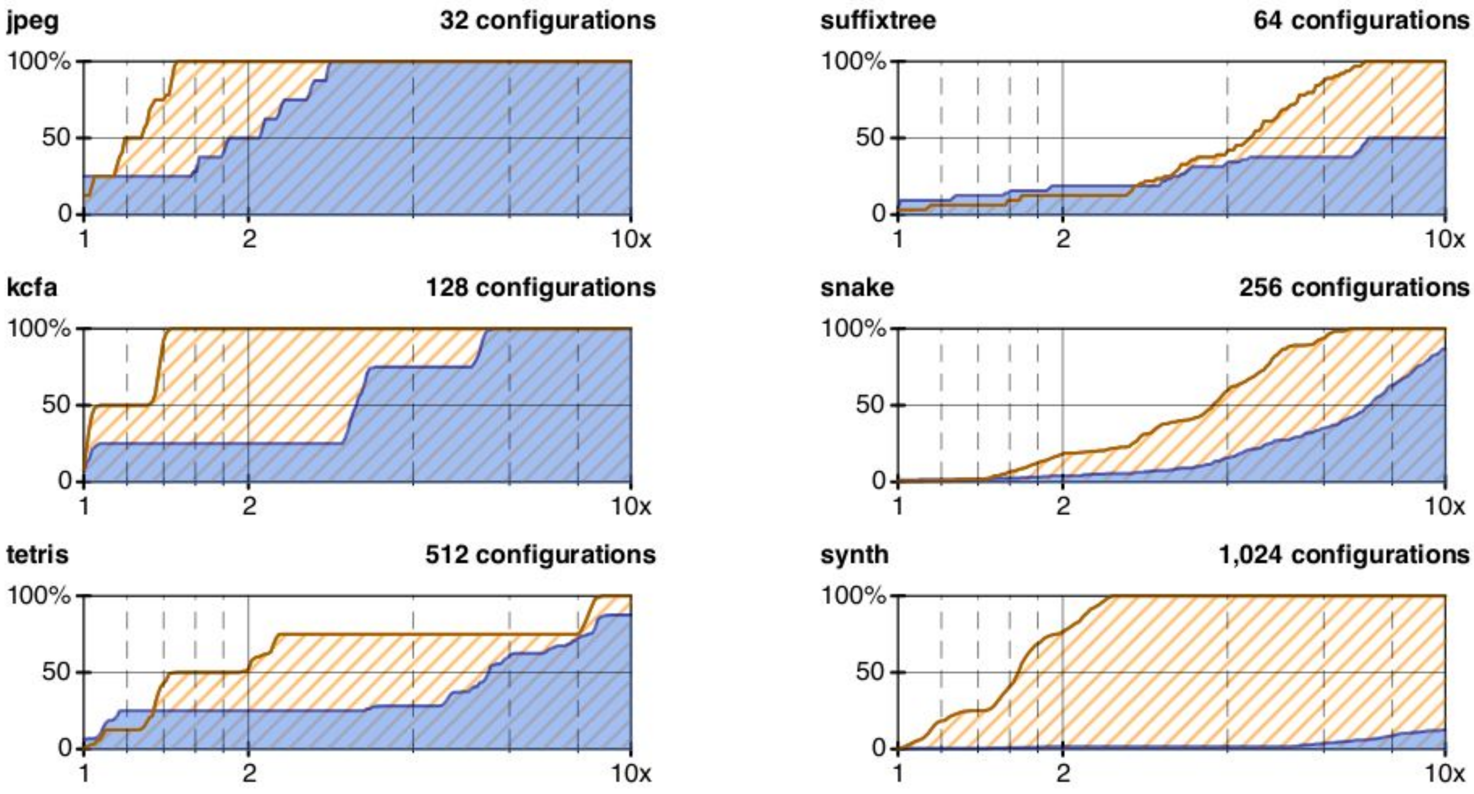
- $e'' \rightarrow^*_{\text{1-S}} v$ and $\vdash_1 v : \lfloor \tau \rfloor$
- $e'' \rightarrow^*_{\text{1-S}} E[\text{dyn } \tau' \; E^\bullet[e']]$ and $e' \triangleright_{\text{1-D}} \text{TagErr}$
- $e'' \rightarrow^*_{\text{1-S}} E[\text{dyn } E^\bullet[e']]$ and $e' \triangleright_{\text{1-D}} \text{TagErr}$
- $e'' \rightarrow^*_{\text{1-S}} \text{BndryErr}$
- e'' diverges

Theorem 2.8 : *dynamic* 1-*soundness*

If $e \in e_D$ and $\vdash e$
then $\vdash e \rightsquigarrow e''$ and $\vdash_1 e''$
and one of the following holds:

- $e'' \rightarrow^*_{\text{1-D}} v$ and $\vdash_1 v$
- $e'' \rightarrow^*_{\text{1-D}} E[e']$ and $e' \triangleright_{\text{1-D}} \text{TagErr}$
- $e'' \rightarrow^*_{\text{1-D}} \text{BndryErr}$
- e'' diverges

Performance



Mixed-Typed Programs

Fig. 10. TR-H (blue) and TR-1 (orange), each relative to erasure (TR-E). The *x*-axis is log-scaled. The unlabeled vertical ticks appear at: 1.2x, 1.4x, 1.6x, 1.8x, 4x, 6x, and 8x overhead. A larger area under the curve is better.

Corner Cases

	sie.	fsm	mor.	zom.	jpeg	suf.	kcfa	sna.	tet.	syn.
TR-H	21.76x	506.10x	2.01x	1072.80x	2.81x	24.59x	5.57x	13.15x	13.93x	51.38x
TR-1	1.69x	1.21x	3.48x	20.36x	1.47x	7.10x	1.44x	6.72x	8.88x	2.49x

Fig. 11. Worst-case overhead for higher-order (TR-H) and first-order (TR-1), each relative to erasure.

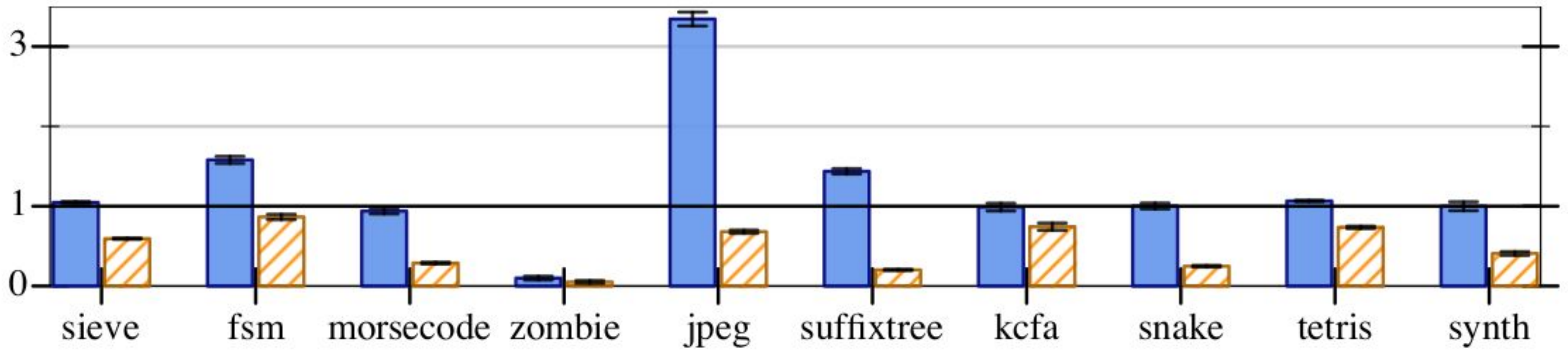


Fig. 12. Speedup of fully-typed TR-H (▢) and TR-1 (▨), relative to TR-E (the 1x line). Taller bars are better than shorter bars.

Takeaways

Mixed-typed configurations:

1. Erasure adds no overhead.
2. Higher-order approach may render a working program **unusably slow**.
3. First-order checks add overhead on a **pay-as-you-annotate** basis.

Fully-typed programs:

- Higher-order embedding often provides the **best performance** of all three.

Threats To Validity

1. Useful error messages are not provided (↑)
2. Translation and optimizer of TR-1 do not ellide redundant checks (↓)
3. TR-1 does not support full Racket (~)
4. Small Benchmarks, <1.5K LOC (?)
5. Single fully-typed version

Implications

For Base Types

- *Erasure*: while maintaining rudimentary soundness, fails to draw distinction between logically different but identically represented values (e.g. Nat vs. Int).

- *Erasure*: in extreme cases of host languages (e.g. JavaScript) it may be hard to get even TagErr. I. e. **Garbage In—Garbage Out**.

- *HO and FO* have water-proof soundness for base types: if `v: Nat` then `v` is a natural number.

For a First-Order, Non-Base Type:
1st order is unable to see through a constructor

✓	$\vdash$dyn (Nat×Nat) ⟨−2,−2⟩: Nat×Nat	$\rightarrow^*_{\text{H-S}}$	BndryErr
⚠	$\vdash$dyn (Nat×Nat) ⟨−2,−2⟩: Nat×Nat	$\rightarrow^*_{\text{1-S}}$	⟨−2,−2⟩
✓	$\vdash$fst (dyn (Nat×Nat) ⟨−2,−2⟩): Nat	$\rightarrow^*_{\text{1-S}}$	BndryErr
⚠	$\vdash$fst (dyn (Nat×Nat) ⟨−2,−2⟩): Int	$\rightarrow^*_{\text{1-S}}$	−2

⇒ a developer cannot assume that a value of type $\tau_1 \times \tau_2$ contains components of type τ_1 and type τ_2 because **type-constructor soundness is not compositional**

For Higher-Order Types: 1st-order can raise TagErr

⚠

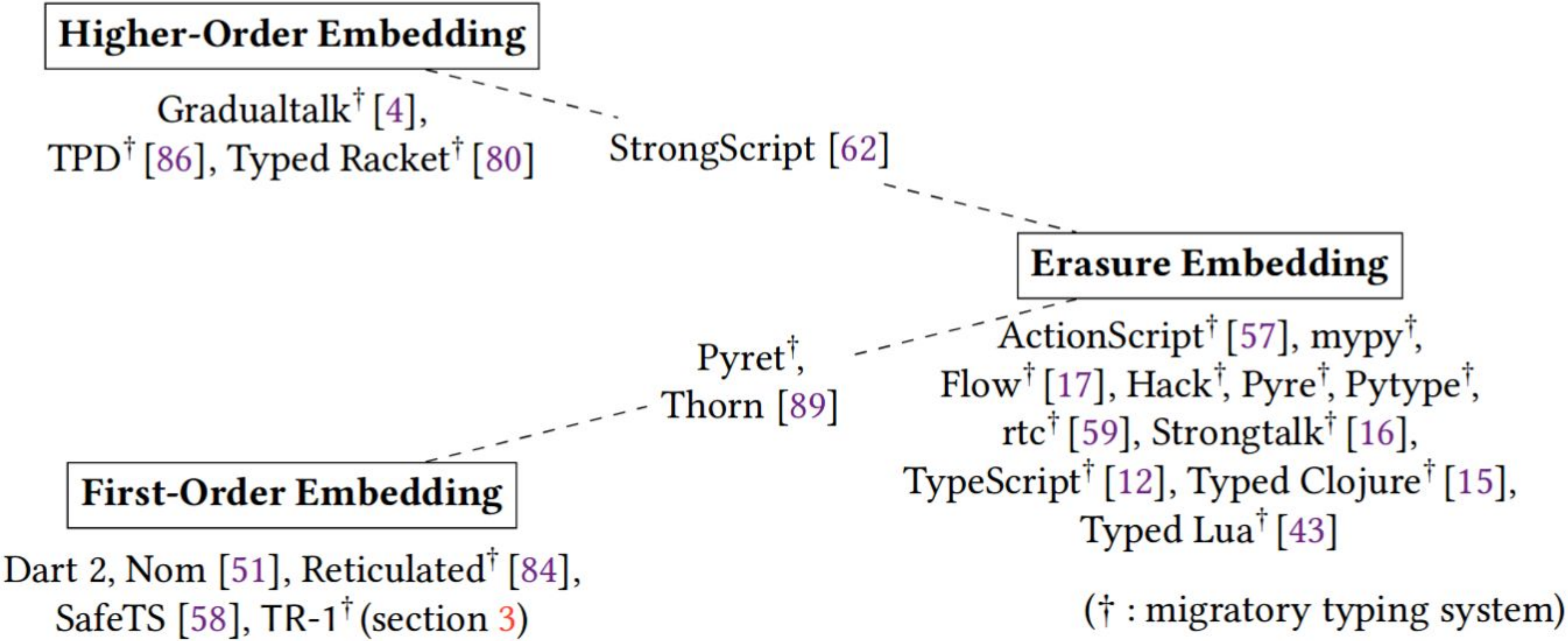
$$f = (\lambda x.\, x\ \langle 1, 1\rangle)$$
$$h = \mathsf{dyn}\ (\mathsf{Nat} \Rightarrow \mathsf{Nat})\ (\lambda y.\, \mathsf{sum}\ y\ y)$$
$$\vdash (\mathsf{dyn}\ ((\mathsf{Nat} \Rightarrow \mathsf{Nat}) \Rightarrow \mathsf{Nat})\ f)\ h : \mathsf{Nat} \rightarrow^*_{\text{1-S}} f\ h \rightarrow^*_{\text{1-S}} h\ \langle 1, 1\rangle \rightarrow^*_{\text{1-S}} \mathsf{TagErr}$$

⚠

$$g = \mathsf{dyn}\ (\mathsf{Int}{\times}\mathsf{Int} \Rightarrow \mathsf{Int})\ (\lambda x.\, \mathsf{snd}\, x)$$
$$\vdash (\mathsf{stat}\ (\mathsf{Int} \Rightarrow \mathsf{Int})\ g)\ 2 \rightarrow^*_{\text{1-D}} \mathsf{snd}\ 2 \rightarrow^*_{\text{1-D}} \mathsf{TagErr}$$

Discussion and Ongoing Work

What’s With “Real-World”



Is Sound Gradual Typing Inevitably Slow?

Not really. Possible approaches (OOPSLA ‘17):

- Use nominal language with concrete types
  (“Sound Gradual Typing Is Nominally Alive and Well”)

- Use JIT-compilers to ellide redundant checks in first-order
  (“Sound Gradual Typing: Only Mostly Dead”)

- Modify regular VM type checks to check shape too
  (“The VM Already Knew That: Leveraging Compile-Time Knowledge to Optimize Gradual Typing”)

¿ what’s in common ?
⇒ all are **first-order** developments!

Besides Performance: Error Messages

- Erasure: gives no clue what went wrong.

- First-Order: reveals a violation of types if it affects the **execution of typed code**.

- Higher-Order: uncovers a violation of type annotations as soon as there is **a witness** and pinpoints **the exact type boundary** that is violated by this witness.

How SW developers feel about error messages?

In

> “The behavior of gradual types: a user study” (DLS 2018)

P. Tunnell Wilson et al. find:

- Erasure feels unexpected and is disliked.
- Higher-Order feels expected and is liked.
- First-Order received mixed feedback.

Fin