A Spectrum of Type Soundness and Performance

Ben Greenman & Matthias Felleisen
Northeastern University

Is type soundness all-or-nothing?

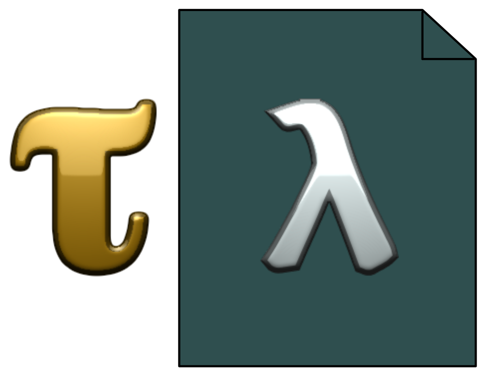
How does type soundness affect performance?

Migratory Typing

Migratory Typing

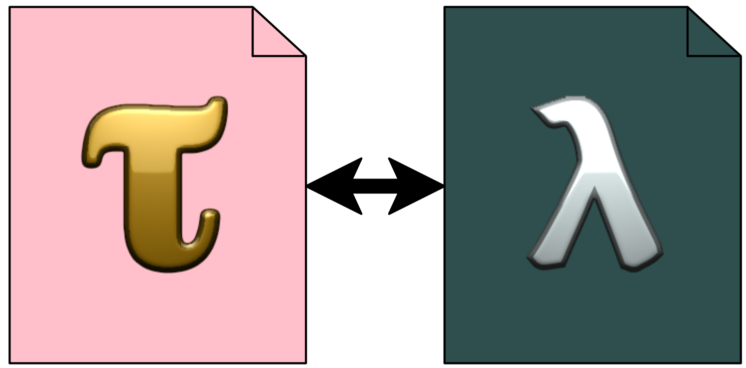
Begin with a un(i)typed language

Migratory Typing



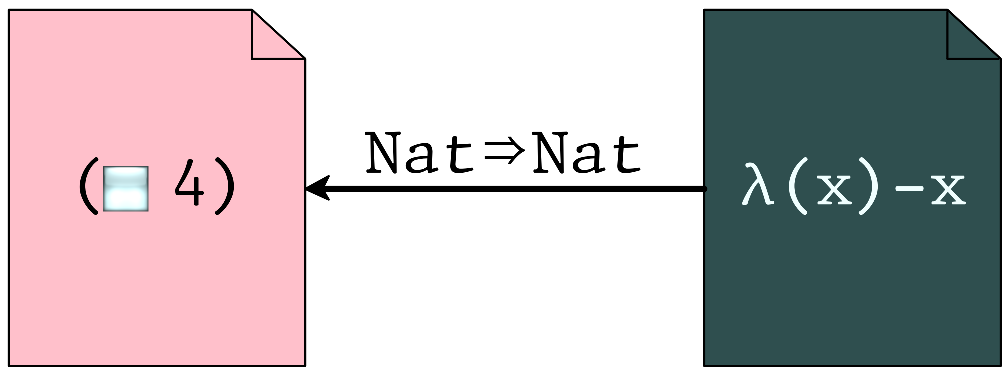
Design an idiomatic type system

Migratory Typing



Result: a mixed-typed language

Migratory Typing



Result: a mixed-typed language

Mixing Typed and Untyped Code

- migratory typing

  gradual typing

  optional typing

  concrete typing

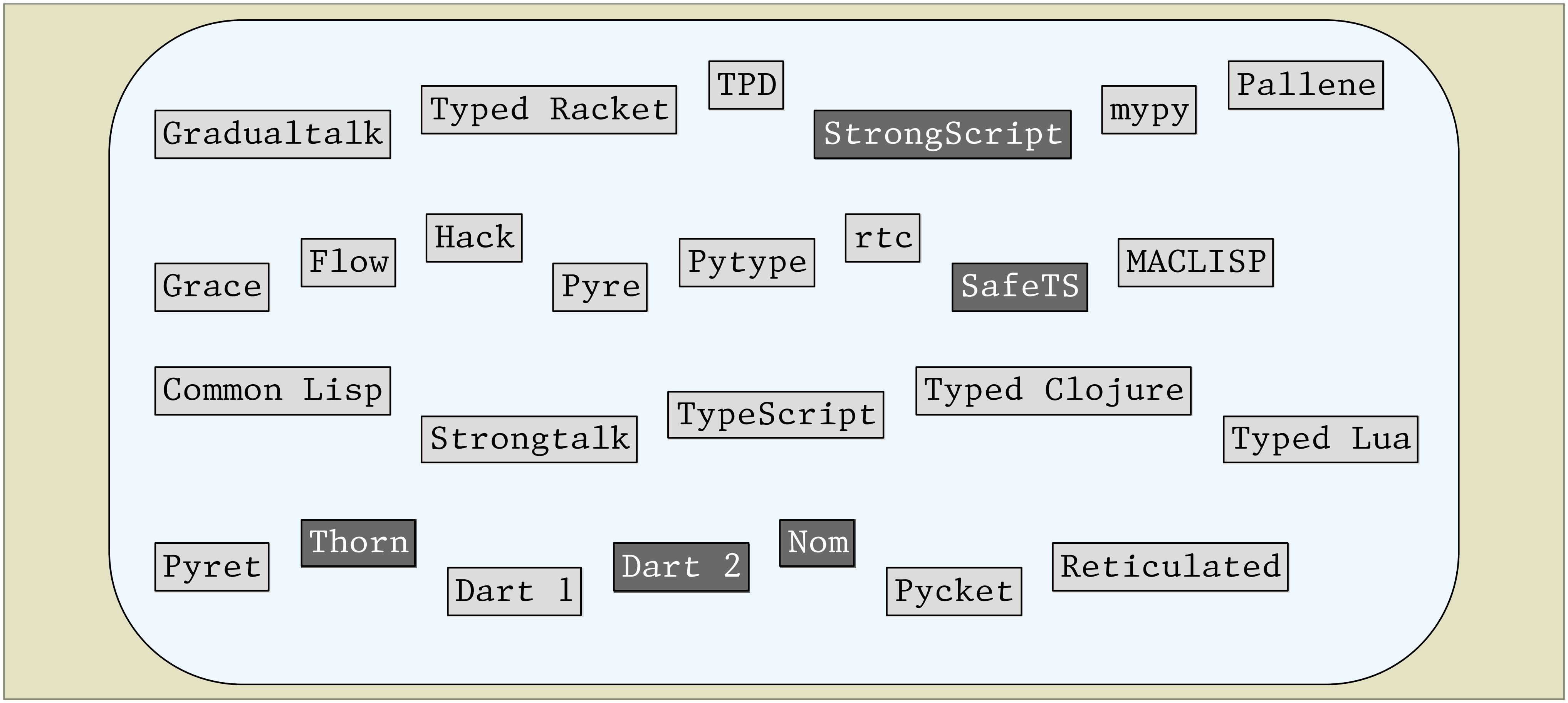
  ...

(the research landscape)

Mixed-Typed Languages

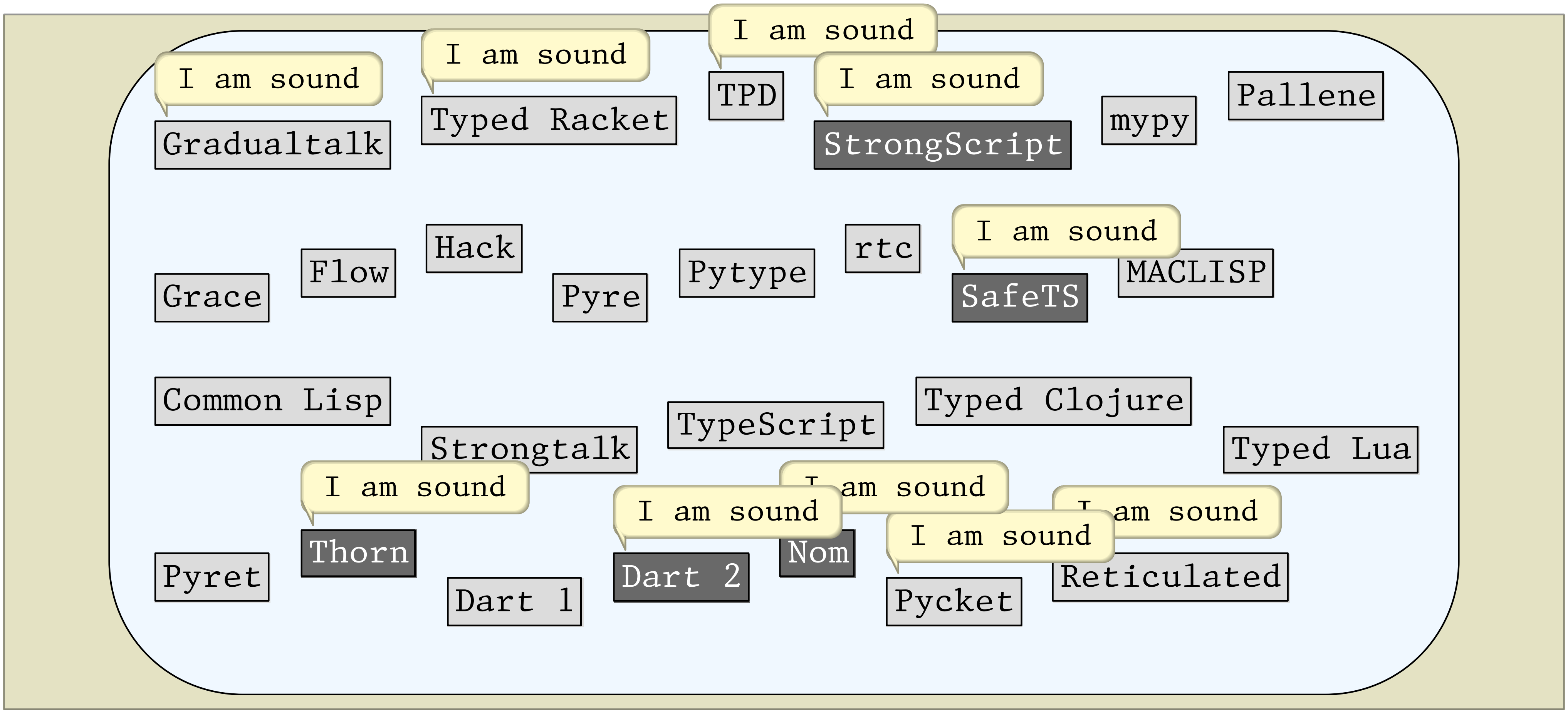
(the systems landscape)

Mixed-Typed Languages



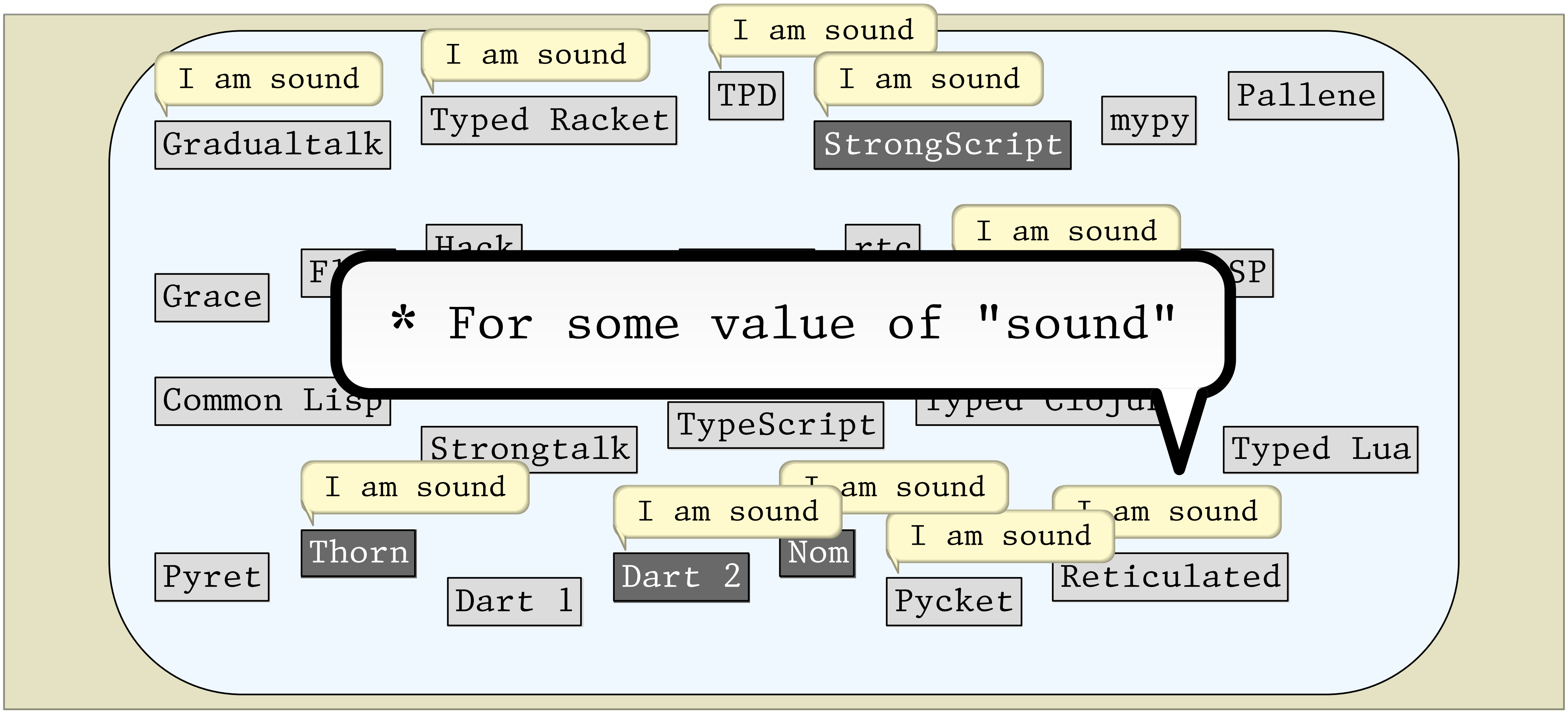
(the systems landscape)

Mixed-Typed Languages



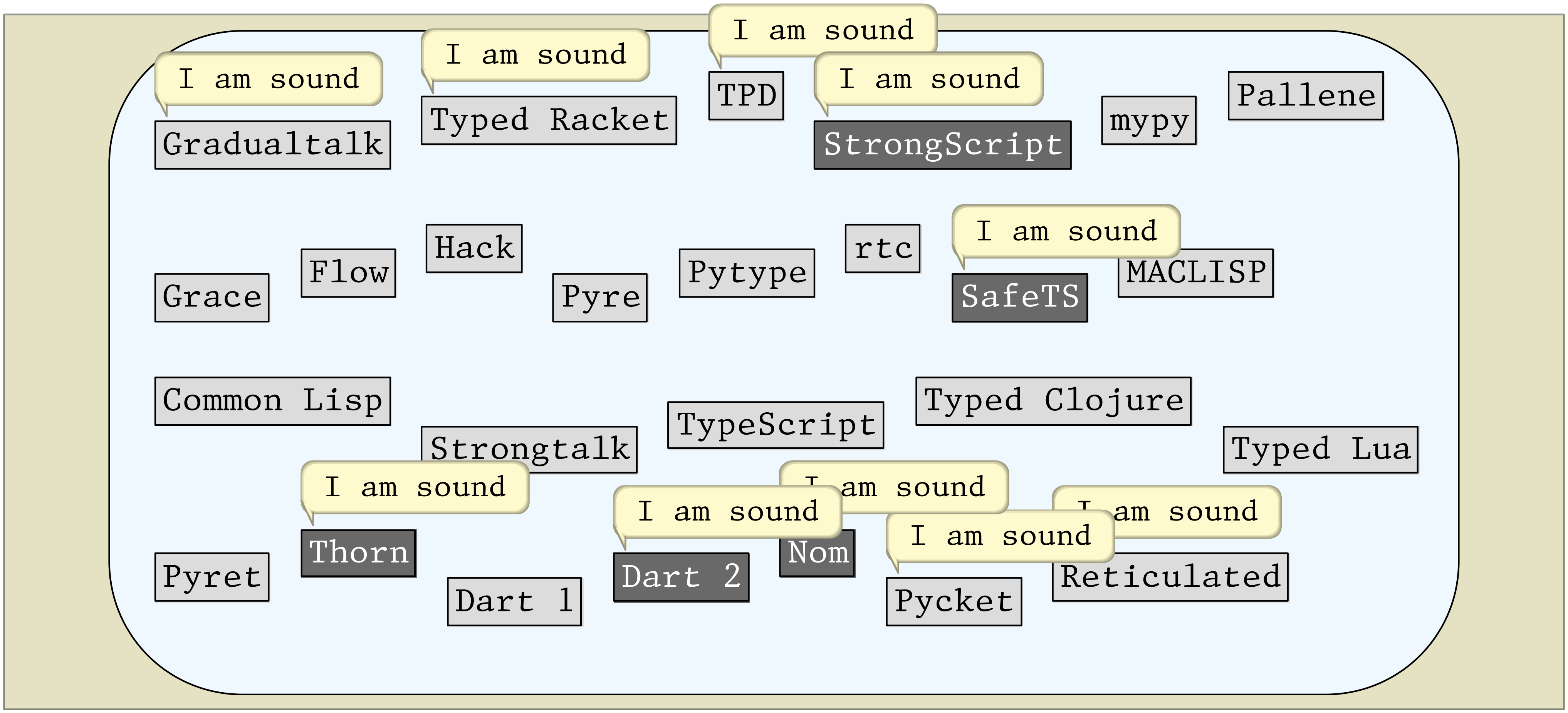
(the systems landscape)

Mixed-Typed Languages



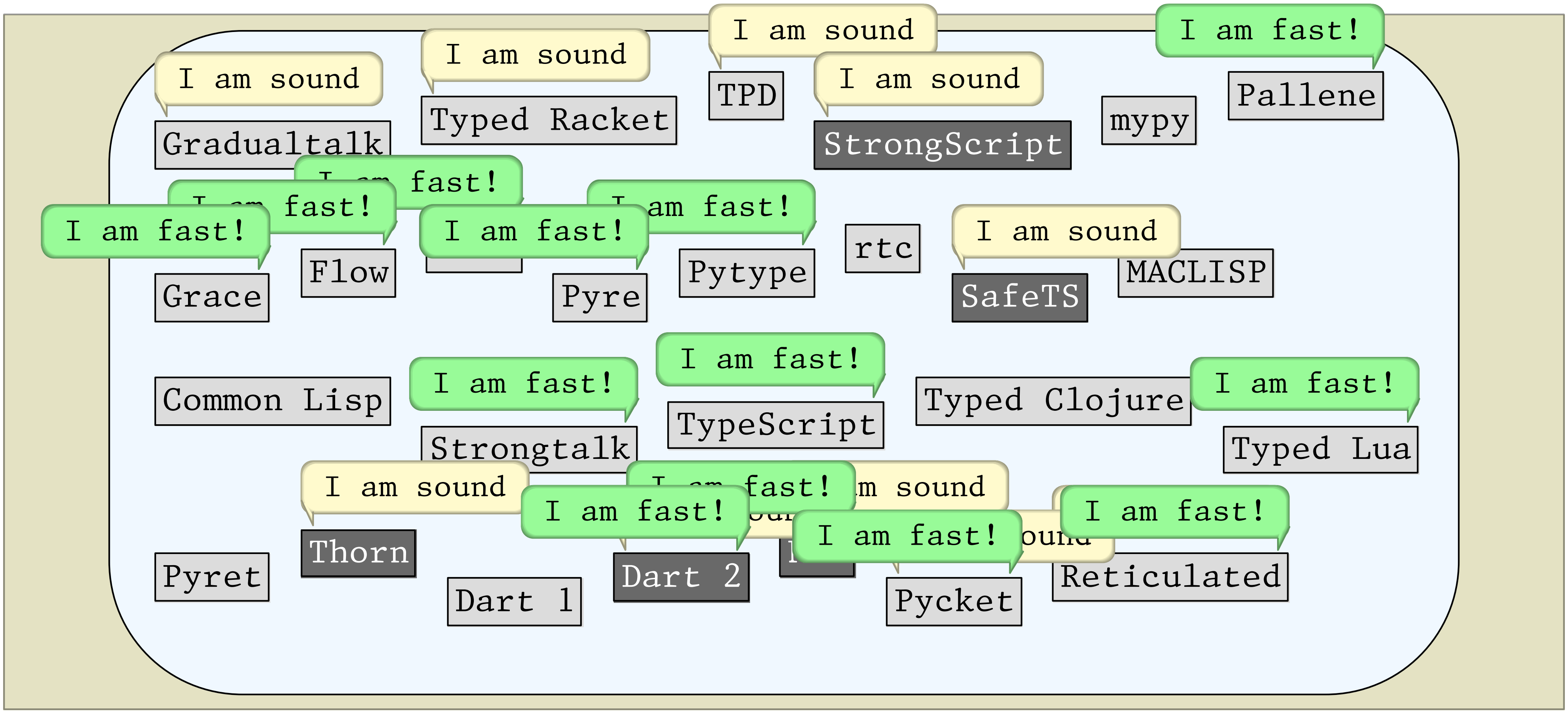
(the systems landscape)

Mixed-Typed Languages



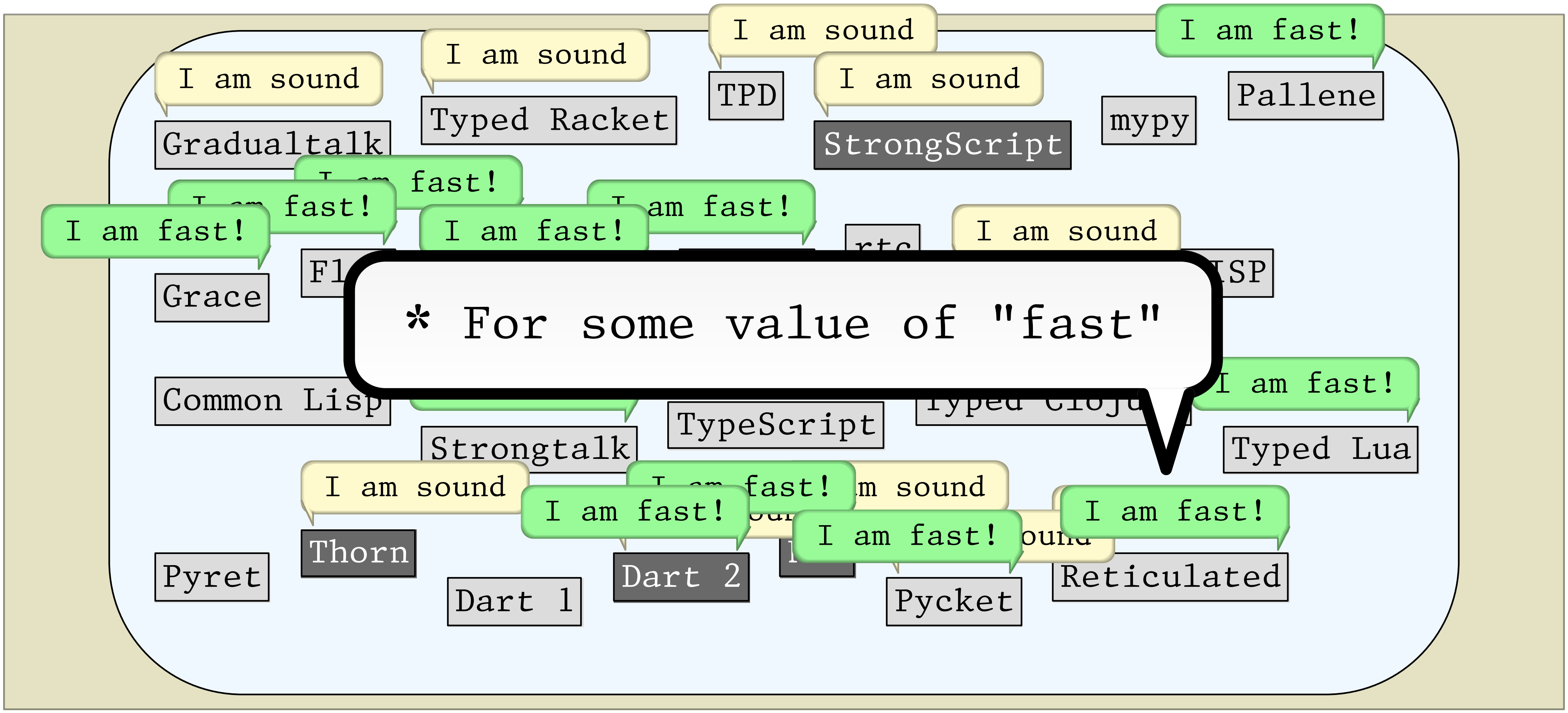
(the systems landscape)

Mixed-Typed Languages



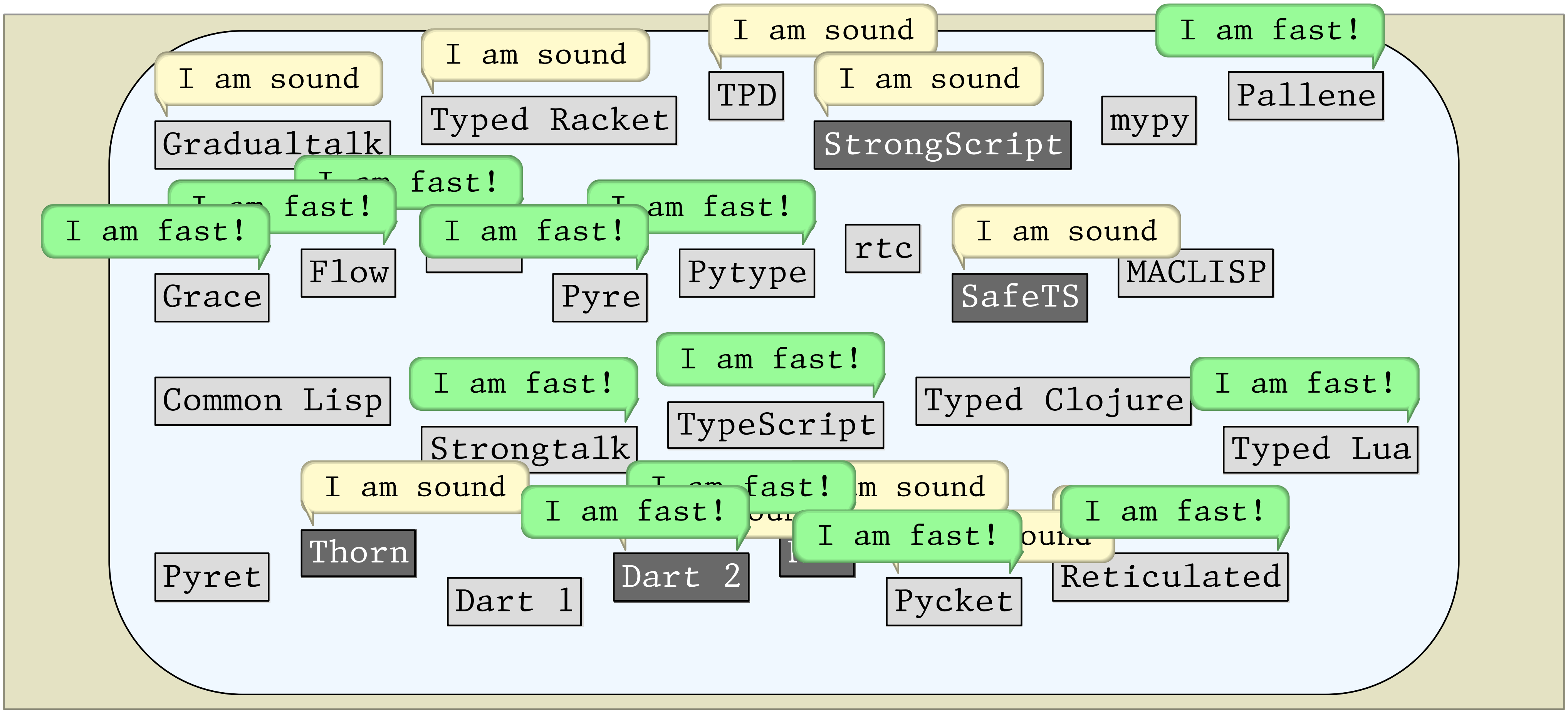
(the systems landscape)

Mixed-Typed Languages



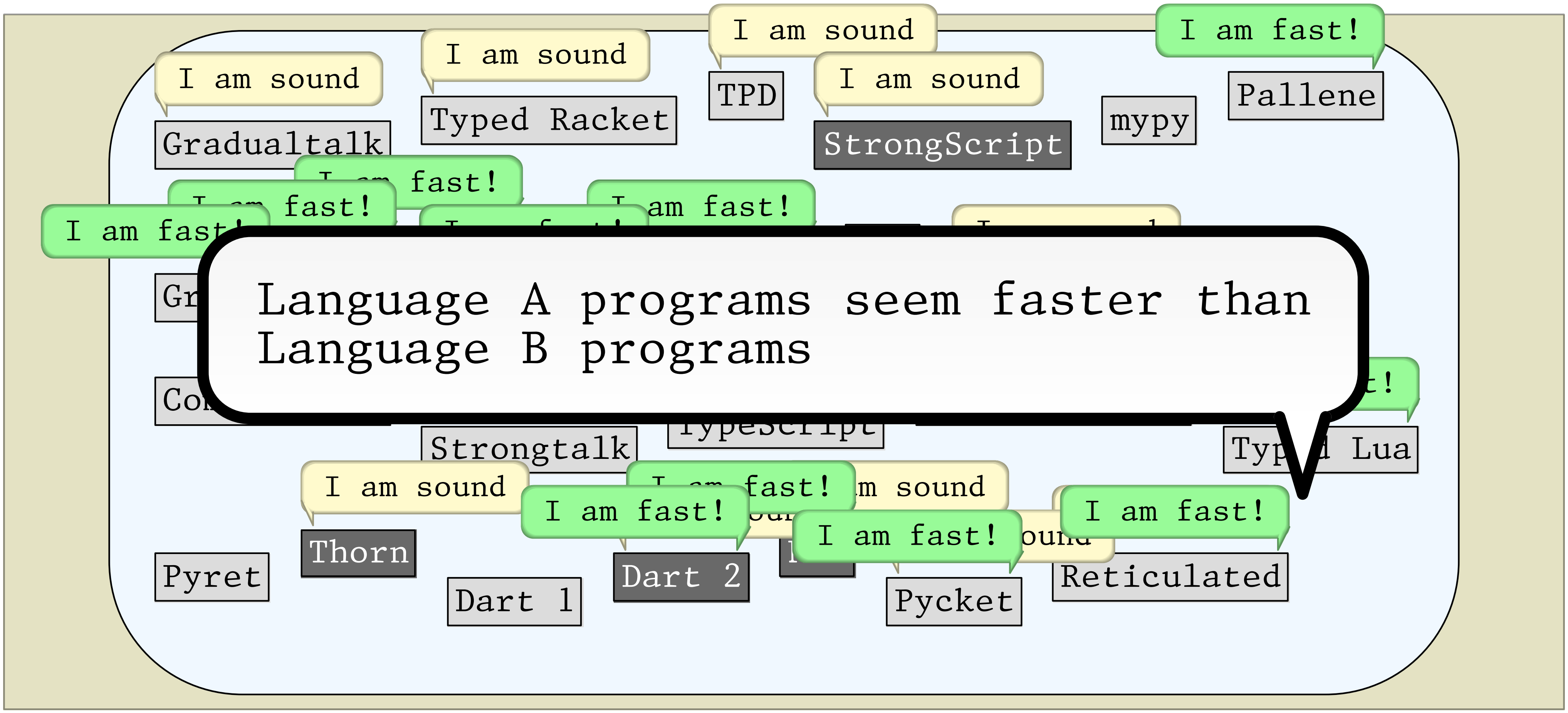
(the systems landscape)

Mixed-Typed Languages



(the systems landscape)

Mixed-Typed Languages

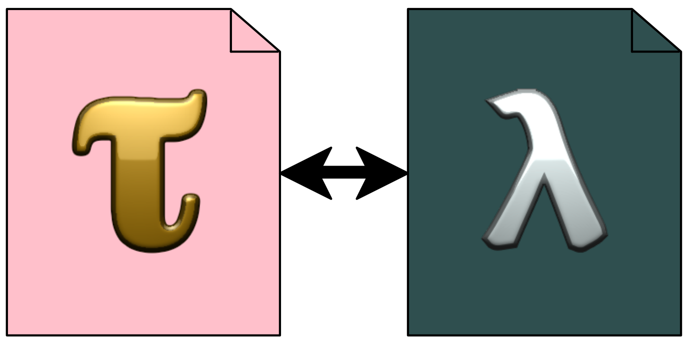


(the systems landscape)

In this paper:

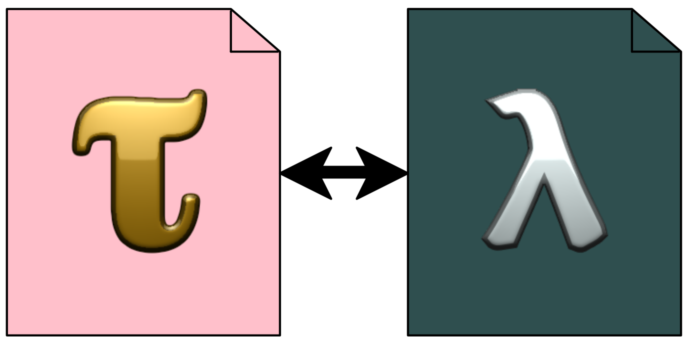
Let's put the **theory** and **practice** on firm scientific ground.

In this paper:



`One mixed-typed language ...`

In this paper:



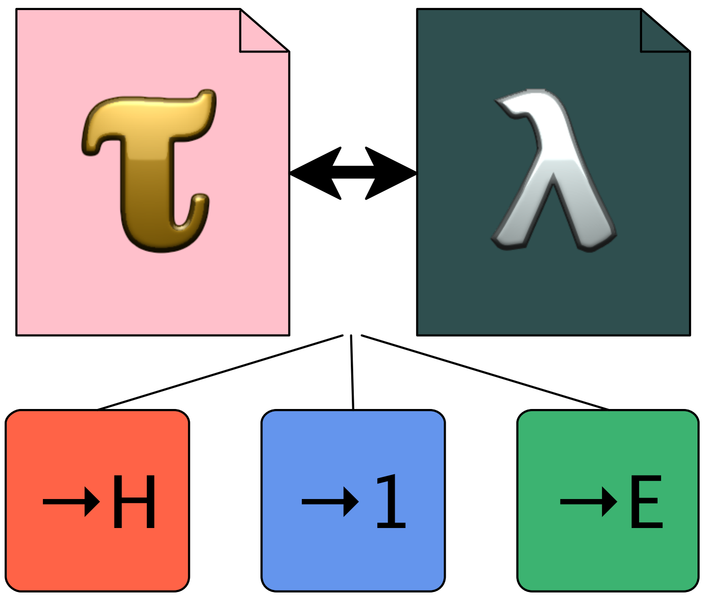
→H higher-order

→1 first-order

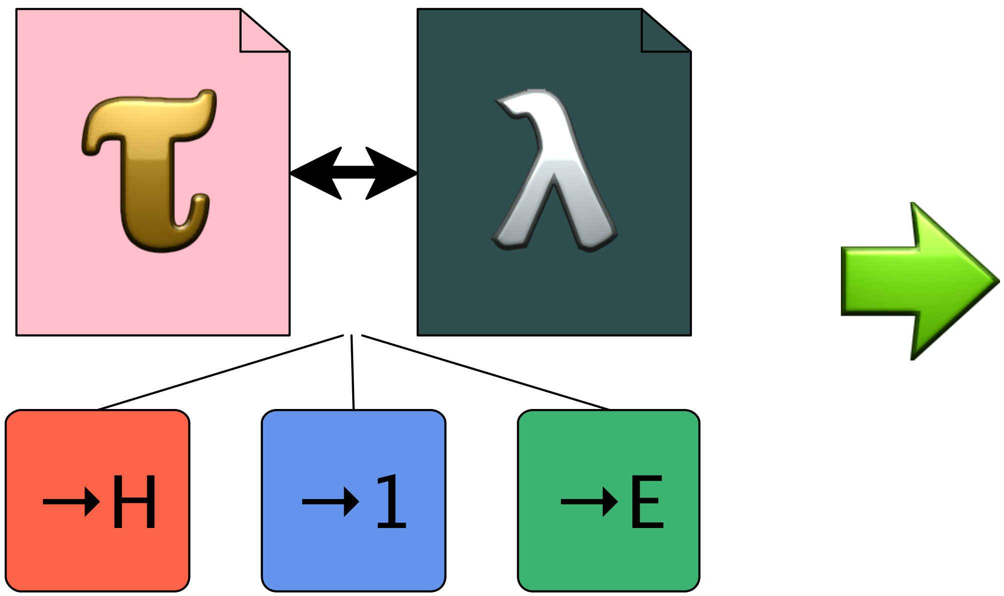
→E erasure

One mixed-typed language ... three semantics

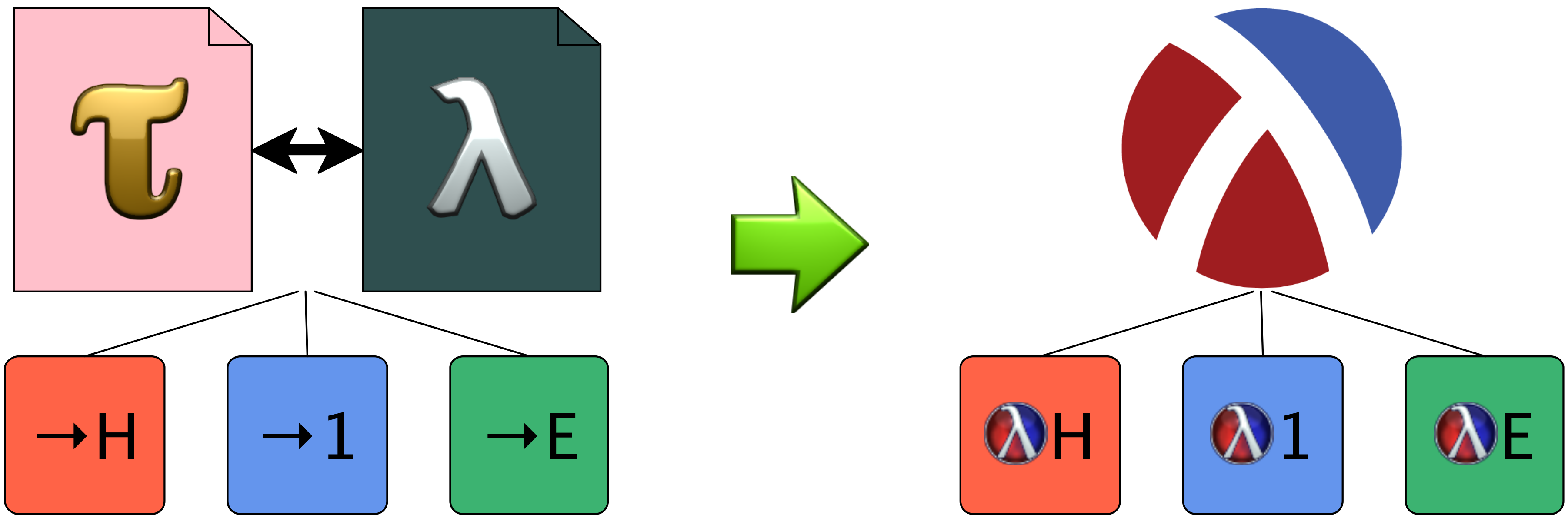
Apples-to-Apples Theory



supports direct comparisons of the meta-theory



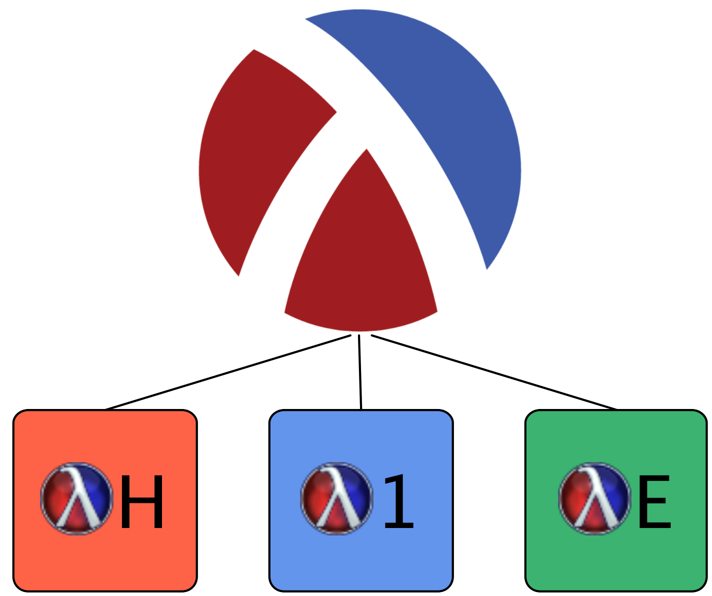
model => implementation



model => implementation

Apples-to-Apples Performance

`able to systematically`
`compare running times`



Model

τ = Nat | Int | $\tau \times \tau$ | $\tau \Rightarrow \tau$

Nat <: Int

τ = Nat | Int | $\tau \times \tau$ | $\tau \Rightarrow \tau$

Nat <: Int

coinductive type

τ = Nat | Int | $\tau \times \tau$ | $\tau \Rightarrow \tau$

Nat <: Int

inductive type

τ = Nat | Int | τ×τ | τ⇒τ

Nat <: Int

base type

$\tau = \mathtt{Nat} \mid \mathtt{Int} \mid \tau \times \tau \mid \tau \Rightarrow \tau$

$\mathtt{Nat} <: \mathtt{Int}$

base type

τ = Nat | Int | $\tau \times \tau$ | $\tau \Rightarrow \tau$

Nat <: Int

subtype relation

τ = Nat | Int | $\tau\times\tau$ | $\tau\Rightarrow\tau$

Nat <: Int

v = n | i | $\langle$v,v$\rangle$ | λ(x)e

n $\subset$ i

τ = Nat | Int | $\tau \times \tau$ | $\tau \Rightarrow \tau$

Nat <: Int

v = n | i | $\langle$v,v$\rangle$ | λ(x)e

n ⊂ i

higher-order value

τ = Nat | Int | $\tau \times \tau$ | $\tau \Rightarrow \tau$

Nat <: Int

v = n | i | ⟨v,v⟩ | λ(x)e

n ⊂ i

data structure

$\tau = \mathtt{Nat} \mid \mathtt{Int} \mid \tau \times \tau \mid \tau \Rightarrow \tau$

$\mathtt{Nat} <: \mathtt{Int}$

$v = n \mid i \mid \langle v, v \rangle \mid \lambda(x)e$

$n \subset i$

base value

$$\tau = \mathtt{Nat} \mid \mathtt{Int} \mid \tau \times \tau \mid \tau \Rightarrow \tau$$

$$\mathtt{Nat} <: \mathtt{Int}$$

$$v = n \mid i \mid \langle v, v \rangle \mid \lambda(x)e$$

$$n \subset i$$

base value

τ = Nat | Int | $\tau \times \tau$ | $\tau \Rightarrow \tau$

Nat <: Int

v = n | i | $\langle$v,v$\rangle$ | λ(x)e

n $\subset$ i

subset relation

τ = Nat | Int | $\tau \times \tau$ | $\tau \Rightarrow \tau$

Nat <: Int

v = n | i | $\langle$v,v$\rangle$ | λ(x)e

n $\subset$ i

e = ... | dyn τ e | stat τ e

τ = Nat | Int | $\tau \times \tau$ | $\tau \Rightarrow \tau$

Nat <: Int

v = n | i | $\langle$v,v$\rangle$ | λ(x)e

n $\subset$ i

e = ... | dyn τ e | stat τ e

boundary terms

τ = Nat | Int | $\tau \times \tau$ | $\tau \Rightarrow \tau$

Nat <: Int

v = n | i | $\langle$v,v$\rangle$ | λ(x)e

n $\subset$ i

e = ... | dyn τ e | stat τ e

$$\tau = \mathrm{Nat} \mid \mathrm{Int} \mid \tau \times \tau \mid \tau \Rightarrow \tau$$

$$\mathrm{Nat} <: \mathrm{Int}$$

$$v = n \mid i \mid \langle v, v \rangle \mid \lambda(x)e$$

$$n \subset i$$

$$e = \ldots. \mid \mathrm{dyn}\ \tau\ e \mid \mathrm{stat}\ \tau\ e$$

$$\boxed{\vdash e : \tau} \qquad \frac{\vdash e}{\vdash \mathrm{dyn}\ \tau\ e : \tau} \qquad\qquad \boxed{\vdash e} \qquad \frac{\vdash e : \tau}{\vdash \mathrm{stat}\ \tau\ e}$$

$$\Gamma = \begin{array}{ll} \texttt{fib} & : \texttt{Nat} \Rightarrow \texttt{Nat} \\ \texttt{norm} & : \texttt{Nat} \times \texttt{Nat} \Rightarrow \texttt{Nat} \\ \texttt{map} & : (\texttt{Nat} \Rightarrow \texttt{Nat}) \Rightarrow \texttt{Nat} \times \texttt{Nat} \Rightarrow \texttt{Nat} \times \texttt{Nat} \end{array}$$

$$\begin{array}{lll} \Gamma \vdash \texttt{fib (dyn Nat -1)} & : & \texttt{Nat} \\ \Gamma \vdash \texttt{norm (dyn Nat} \times \texttt{Nat} \ \langle\texttt{-1,-2}\rangle\texttt{)} & : & \texttt{Nat} \\ \Gamma \vdash \texttt{map (dyn (Nat} \Rightarrow \texttt{Nat) (}\lambda\texttt{(x)-x)) y} & : & \texttt{Nat} \times \texttt{Nat} \end{array}$$

$$\Gamma = \begin{array}{lll} \texttt{fib} & : & \texttt{Nat} \Rightarrow \texttt{Nat} \\ \texttt{norm} & : & \texttt{Nat} \times \texttt{Nat} \Rightarrow \texttt{Nat} \\ \texttt{map} & : & (\texttt{Nat} \Rightarrow \texttt{Nat}) \Rightarrow \texttt{Nat} \times \texttt{Nat} \Rightarrow \texttt{Nat} \times \texttt{Nat} \end{array}$$

$$\begin{array}{lll} \Gamma \vdash \texttt{fib (dyn Nat -1)} & : & \texttt{Nat} \\ \Gamma \vdash \texttt{norm (dyn Nat} \times \texttt{Nat} \langle \texttt{-1,-2} \rangle \texttt{)} & : & \texttt{Nat} \\ \Gamma \vdash \texttt{map (dyn (Nat} \Rightarrow \texttt{Nat) (}\lambda\texttt{(x)-x)) y} & : & \texttt{Nat} \times \texttt{Nat} \end{array}$$

$$\Gamma = \begin{array}{lll} \texttt{fib} & : & \texttt{Nat} \Rightarrow \texttt{Nat} \\ \texttt{norm} & : & \texttt{Nat} \times \texttt{Nat} \Rightarrow \texttt{Nat} \\ \texttt{map} & : & (\texttt{Nat} \Rightarrow \texttt{Nat}) \Rightarrow \texttt{Nat} \times \texttt{Nat} \Rightarrow \texttt{Nat} \times \texttt{Nat} \end{array}$$

$$\begin{array}{lll} \Gamma \vdash \texttt{fib (dyn Nat -1)} & : & \texttt{Nat} \\ \Gamma \vdash \texttt{norm (dyn Nat} \times \texttt{Nat} \langle\texttt{-1,-2}\rangle\texttt{)} & : & \texttt{Nat} \\ \Gamma \vdash \texttt{map (dyn (Nat} \Rightarrow \texttt{Nat) (}\lambda\texttt{(x)-x)) y} & : & \texttt{Nat} \times \texttt{Nat} \end{array}$$

$$\Gamma = \begin{array}{lll} \texttt{fib} & : & \texttt{Nat} \Rightarrow \texttt{Nat} \\ \texttt{norm} & : & \texttt{Nat} \times \texttt{Nat} \Rightarrow \texttt{Nat} \\ \texttt{map} & : & (\texttt{Nat} \Rightarrow \texttt{Nat}) \Rightarrow \texttt{Nat} \times \texttt{Nat} \Rightarrow \texttt{Nat} \times \texttt{Nat} \end{array}$$

$$\begin{array}{lll} \Gamma \vdash \texttt{fib (dyn Nat -1)} & : & \texttt{Nat} \\ \Gamma \vdash \texttt{norm (dyn Nat} \times \texttt{Nat} \langle \texttt{-1,-2} \rangle \texttt{)} & : & \texttt{Nat} \\ \Gamma \vdash \texttt{map (dyn (Nat} \Rightarrow \texttt{Nat) (}\lambda\texttt{(x)-x)) y} & : & \texttt{Nat} \times \texttt{Nat} \end{array}$$

$$\Gamma = \begin{array}{lll} \texttt{fib} & : & \texttt{Nat} \Rightarrow \texttt{Nat} \\ \texttt{norm} & : & \texttt{Nat} \times \texttt{Nat} \Rightarrow \texttt{Nat} \\ \texttt{map} & : & (\texttt{Nat} \Rightarrow \texttt{Nat}) \Rightarrow \texttt{Nat} \times \texttt{Nat} \Rightarrow \texttt{Nat} \times \texttt{Nat} \end{array}$$

$$\begin{array}{lll} \Gamma \vdash \texttt{fib}\ (\texttt{dyn Nat -1}) & : & \texttt{Nat} \\ \Gamma \vdash \texttt{norm}\ (\texttt{dyn Nat} \times \texttt{Nat}\ \langle\texttt{-1,-2}\rangle) & : & \texttt{Nat} \\ \Gamma \vdash \texttt{map}\ (\texttt{dyn}\ (\texttt{Nat} \Rightarrow \texttt{Nat})\ (\lambda(\texttt{x})\texttt{-x}))\ \texttt{y} & : & \texttt{Nat} \times \texttt{Nat} \end{array}$$

$$\Gamma = \begin{array}{lll} \texttt{fib} & : & \texttt{Nat} \Rightarrow \texttt{Nat} \\ \texttt{norm} & : & \texttt{Nat} \times \texttt{Nat} \Rightarrow \texttt{Nat} \\ \texttt{map} & : & (\texttt{Nat} \Rightarrow \texttt{Nat}) \Rightarrow \texttt{Nat} \times \texttt{Nat} \Rightarrow \texttt{Nat} \times \texttt{Nat} \end{array}$$

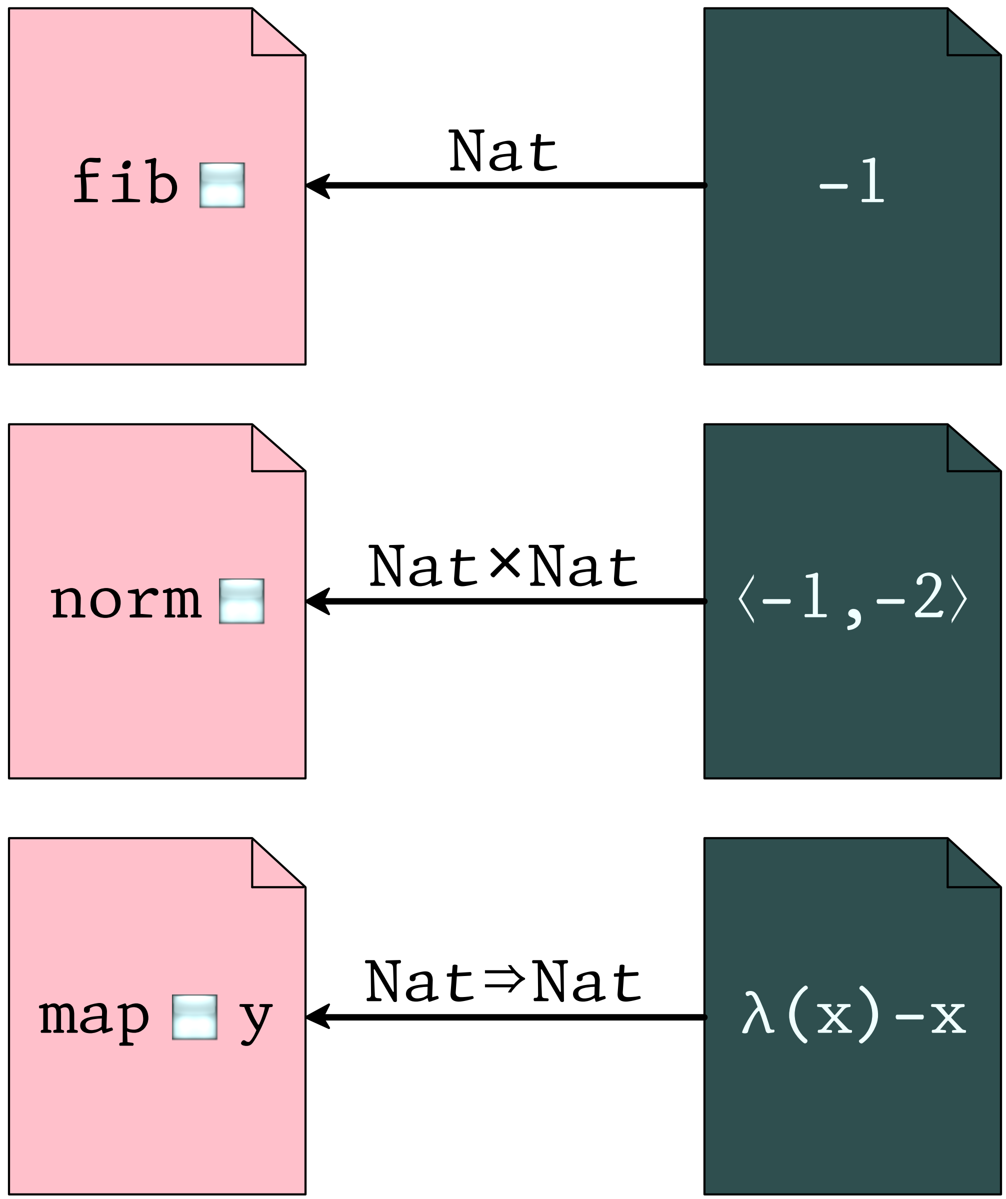
$$\Gamma \vdash \texttt{fib}\ (\texttt{dyn Nat -1}) : \texttt{Nat}$$

$$\Gamma \vdash \texttt{norm}\ (\texttt{dyn Nat} \times \texttt{Nat}\ \langle\texttt{-1,-2}\rangle) : \texttt{Nat}$$

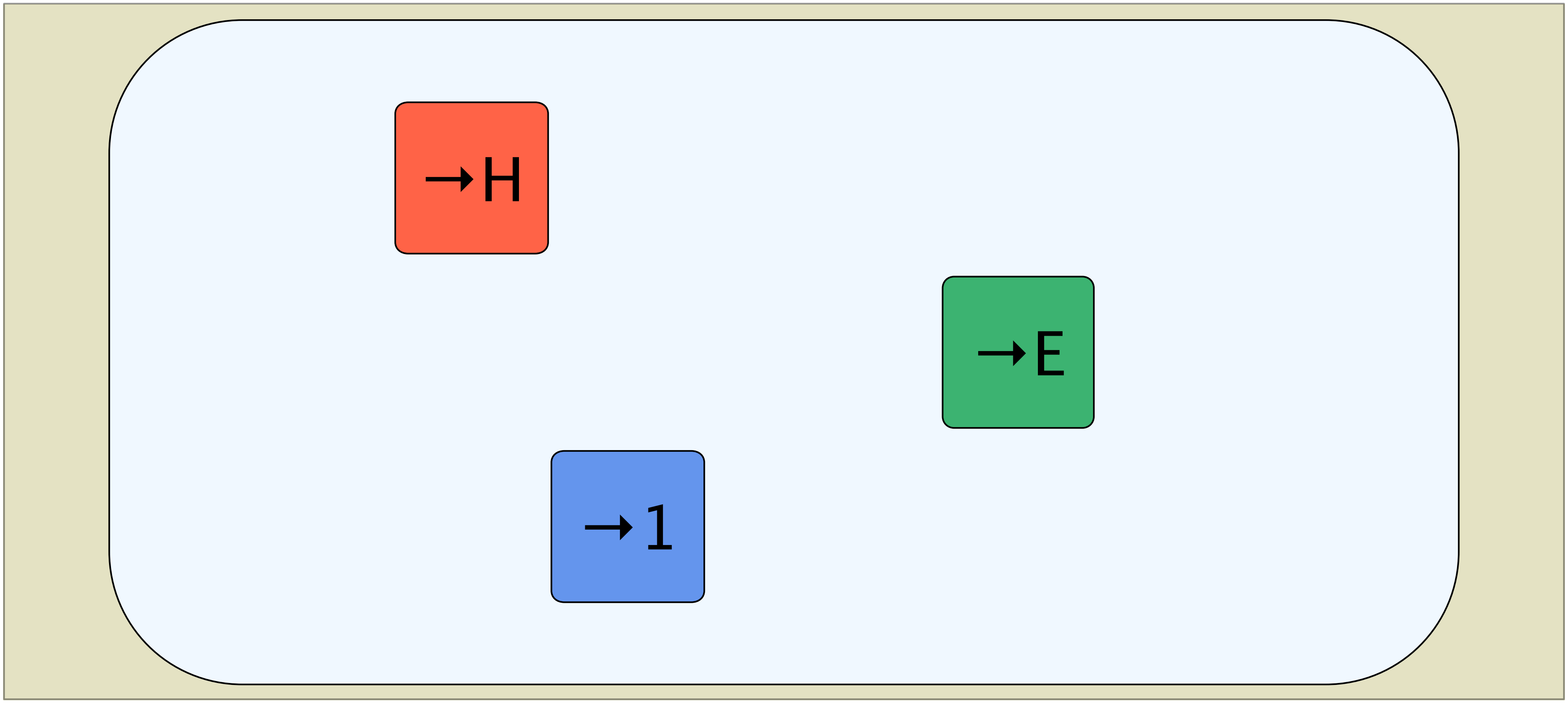
$$\Gamma \vdash \texttt{map}\ (\texttt{dyn}\ (\texttt{Nat} \Rightarrow \texttt{Nat})\ (\lambda(\texttt{x})\texttt{-x}))\ \texttt{y} : \texttt{Nat} \times \texttt{Nat}$$

$$\Gamma = \begin{array}{lll} \texttt{fib} & : & \texttt{Nat} \Rightarrow \texttt{Nat} \\ \texttt{norm} & : & \texttt{Nat} \times \texttt{Nat} \Rightarrow \texttt{Nat} \\ \texttt{map} & : & (\texttt{Nat} \Rightarrow \texttt{Nat}) \Rightarrow \texttt{Nat} \times \texttt{Nat} \Rightarrow \texttt{Nat} \times \texttt{Nat} \end{array}$$

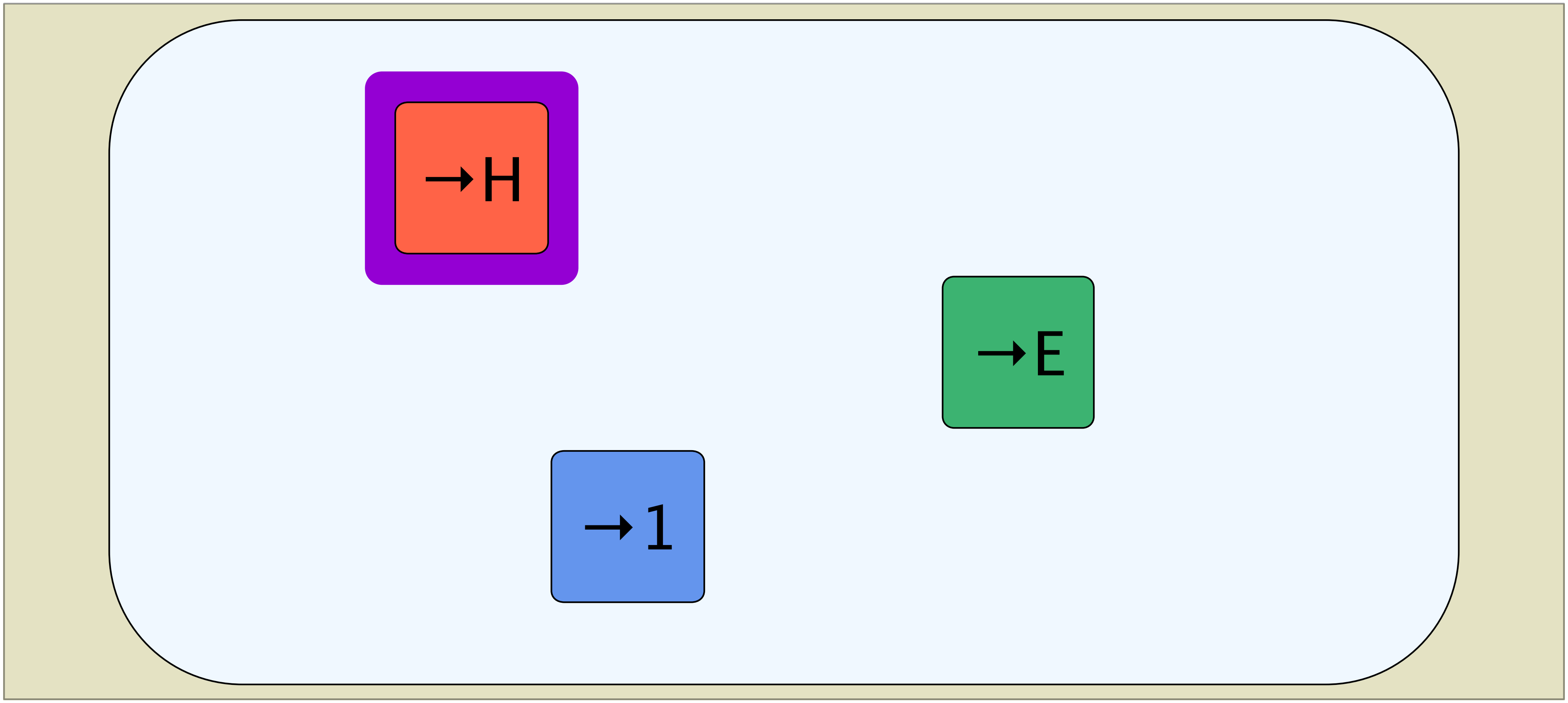
$$\begin{array}{llll} \Gamma \vdash \texttt{fib} & \underline{(\texttt{dyn Nat -1})} & : & \texttt{Nat} \\ \Gamma \vdash \texttt{norm} & \underline{(\texttt{dyn Nat} \times \texttt{Nat} \langle\texttt{-1,-2}\rangle)} & : & \texttt{Nat} \\ \Gamma \vdash \texttt{map} & \underline{(\texttt{dyn (Nat} \Rightarrow \texttt{Nat) (}\lambda\texttt{(x)-x))}}\ \texttt{y} & : & \texttt{Nat} \times \texttt{Nat} \end{array}$$



fib
Nat
-1
norm
Nat×Nat
⟨-1,-2⟩
map y
Nat⇒Nat
λ(x)-x

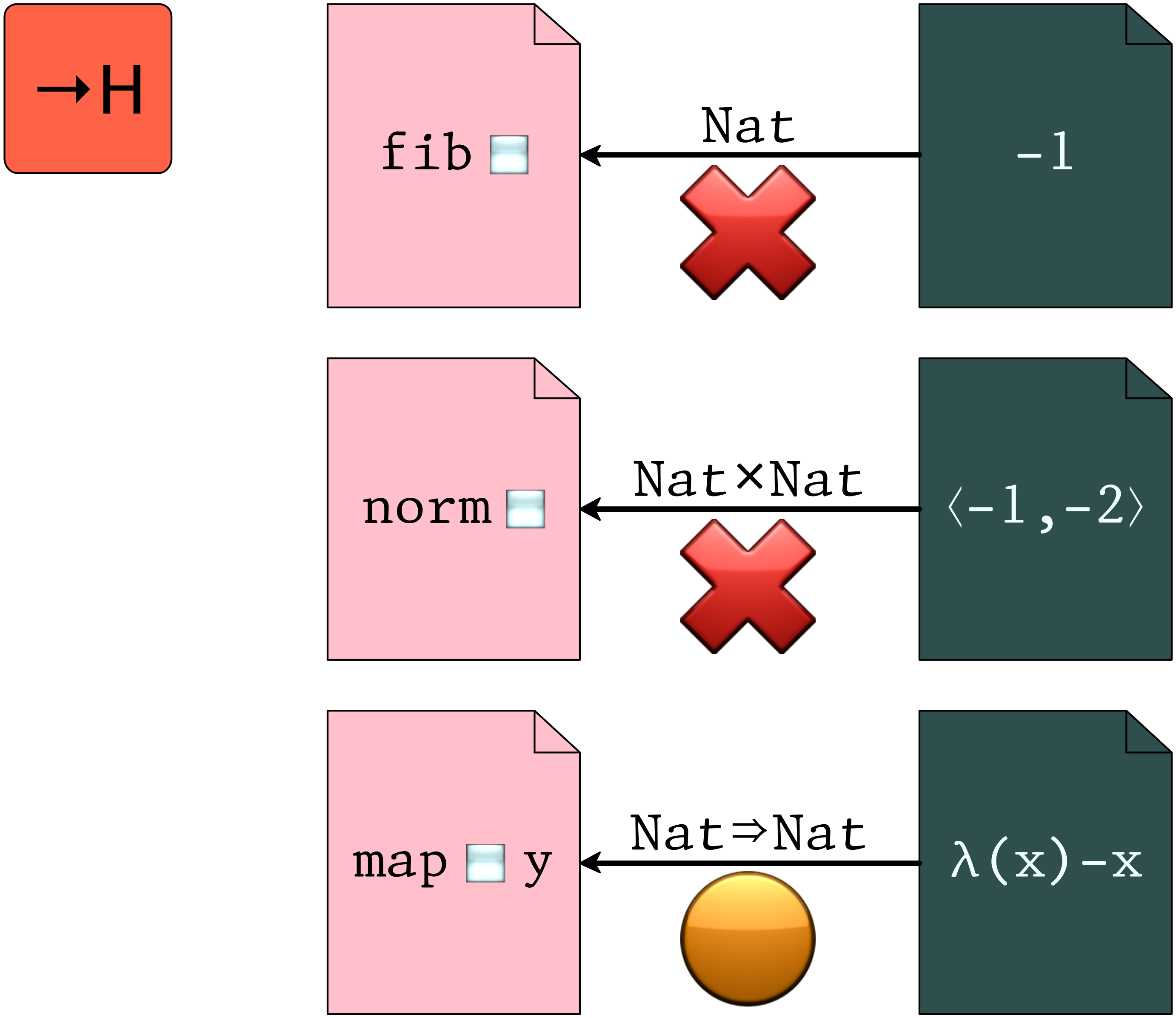


(the systems landscape)

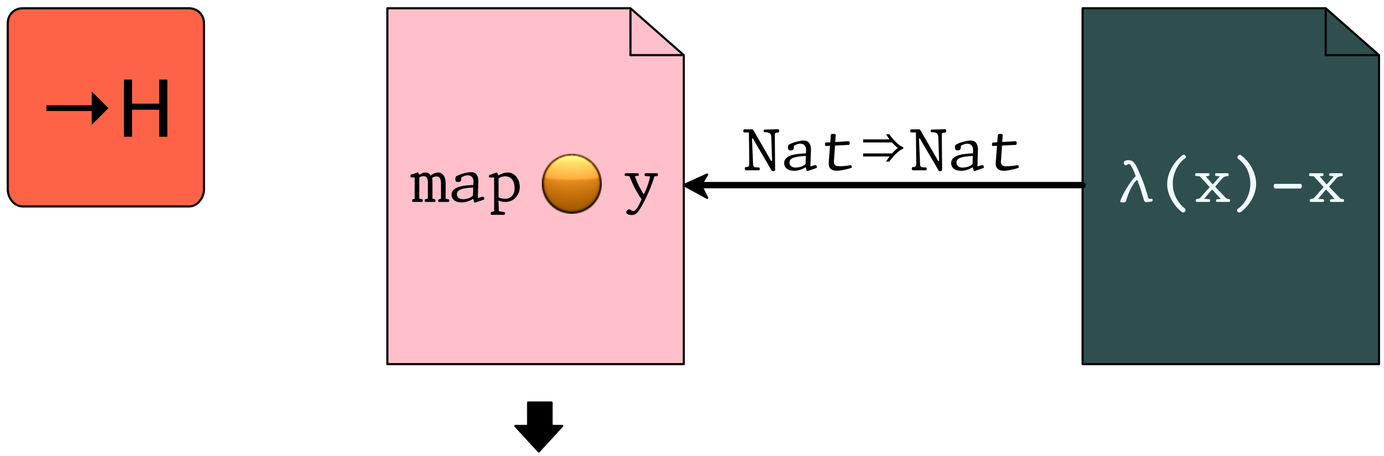


(the systems landscape)

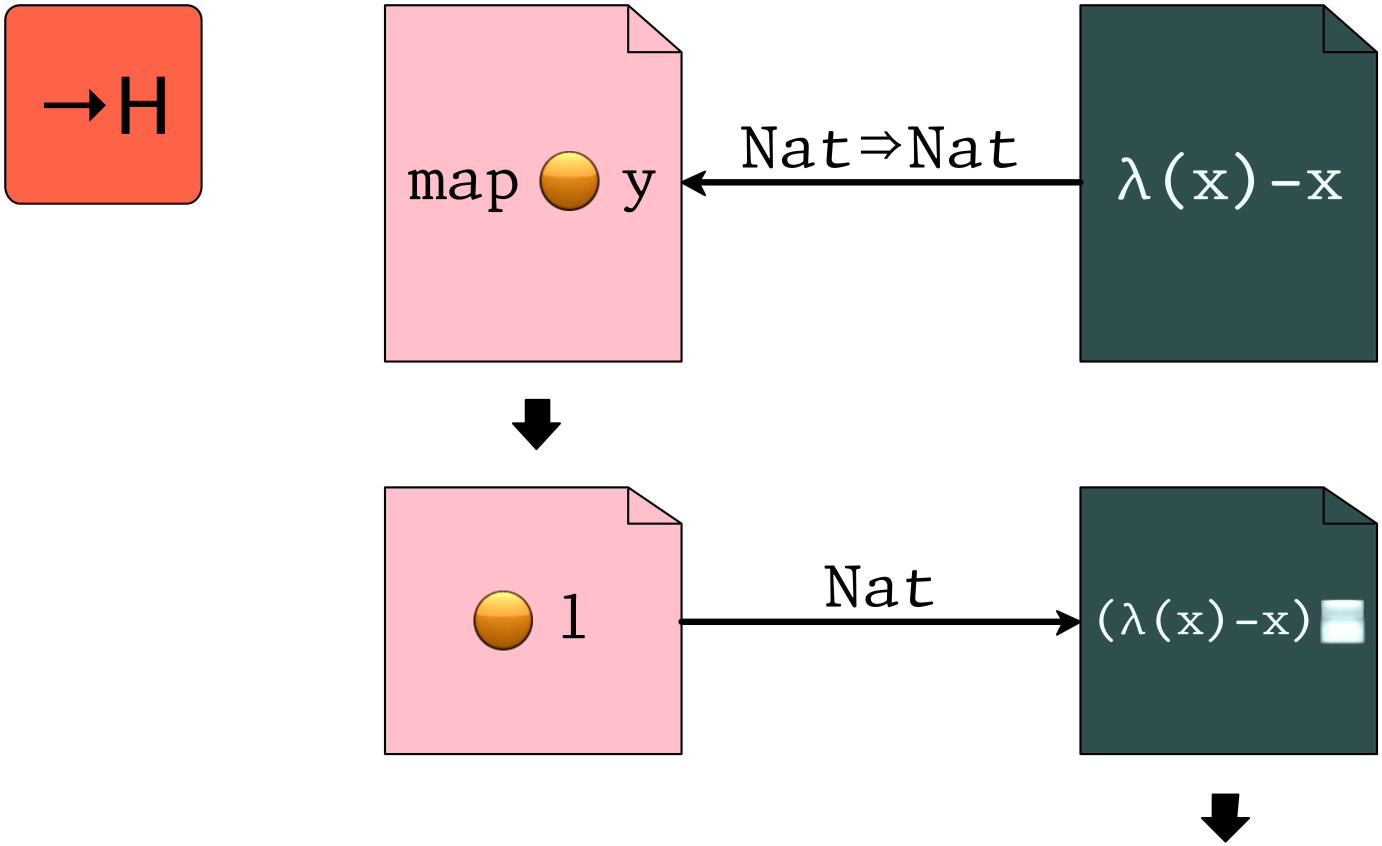
higher-order (enforce full types)



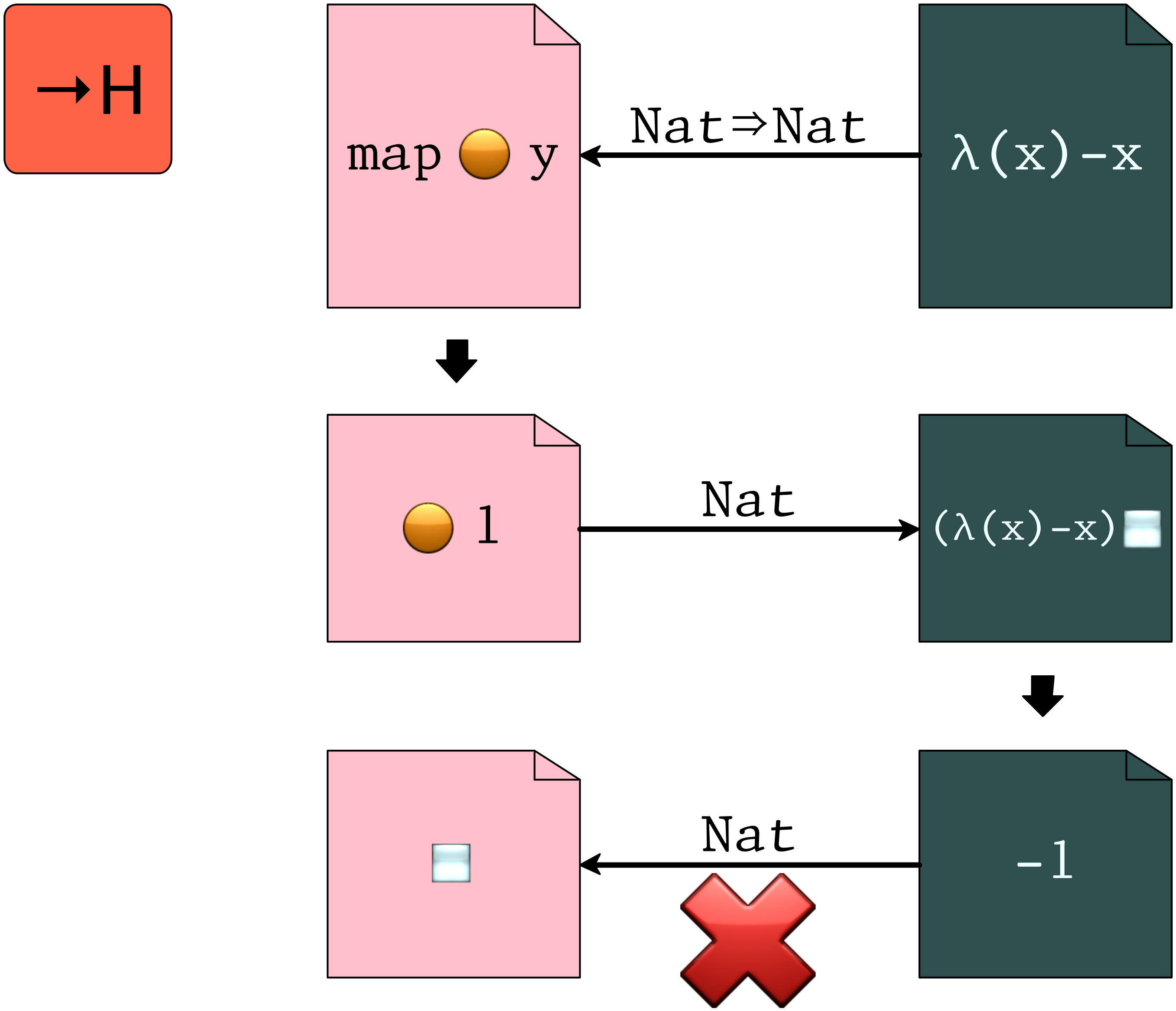
higher-order (enforce full types)

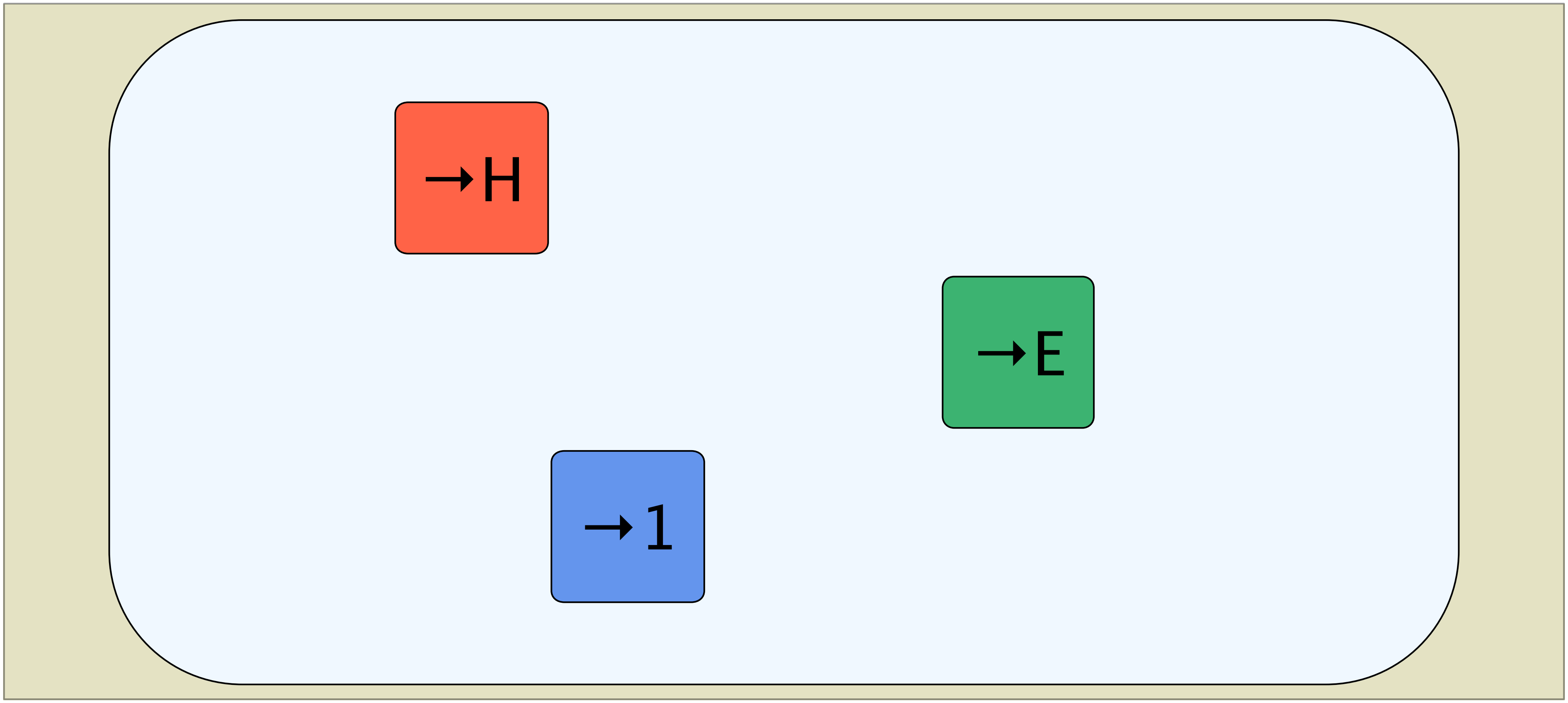


higher-order (enforce full types)

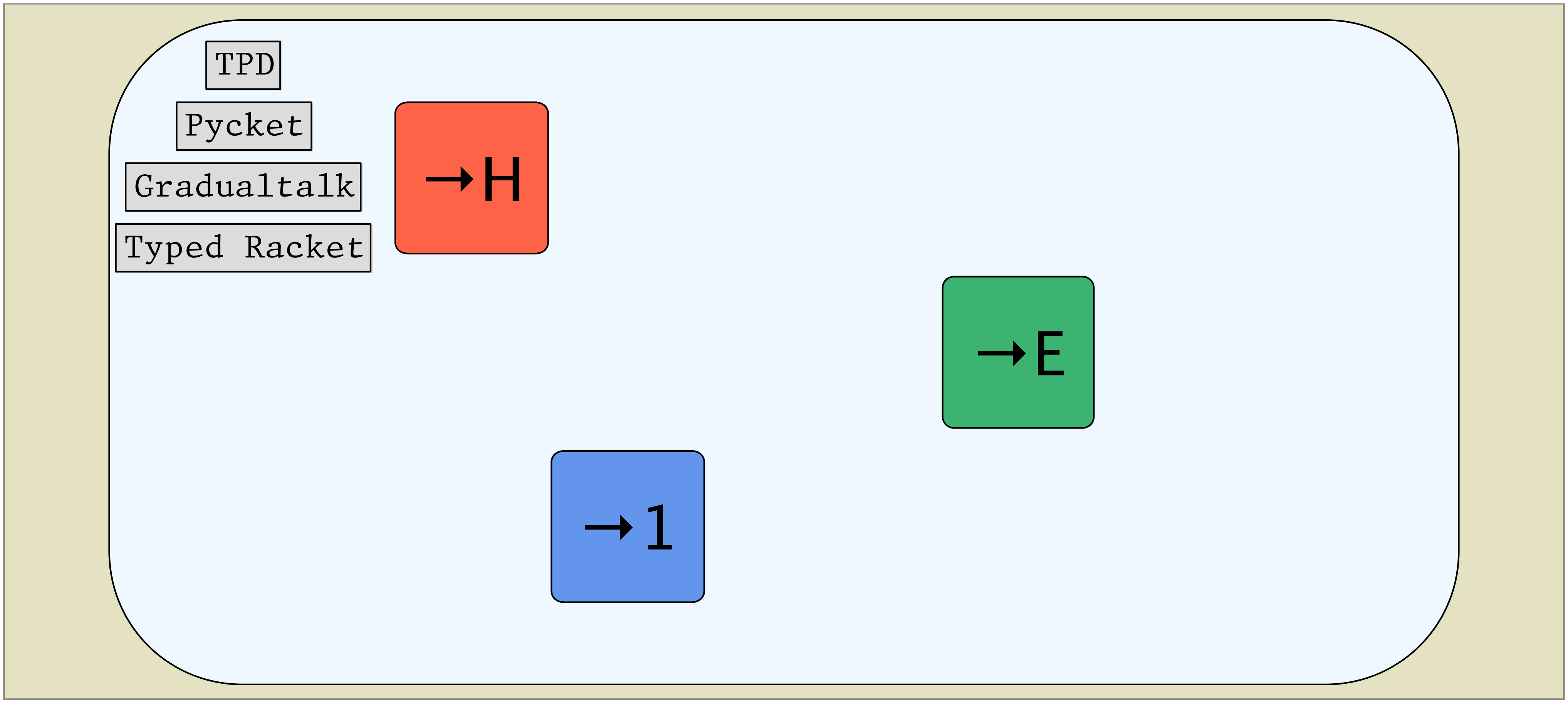


higher-order (enforce full types)

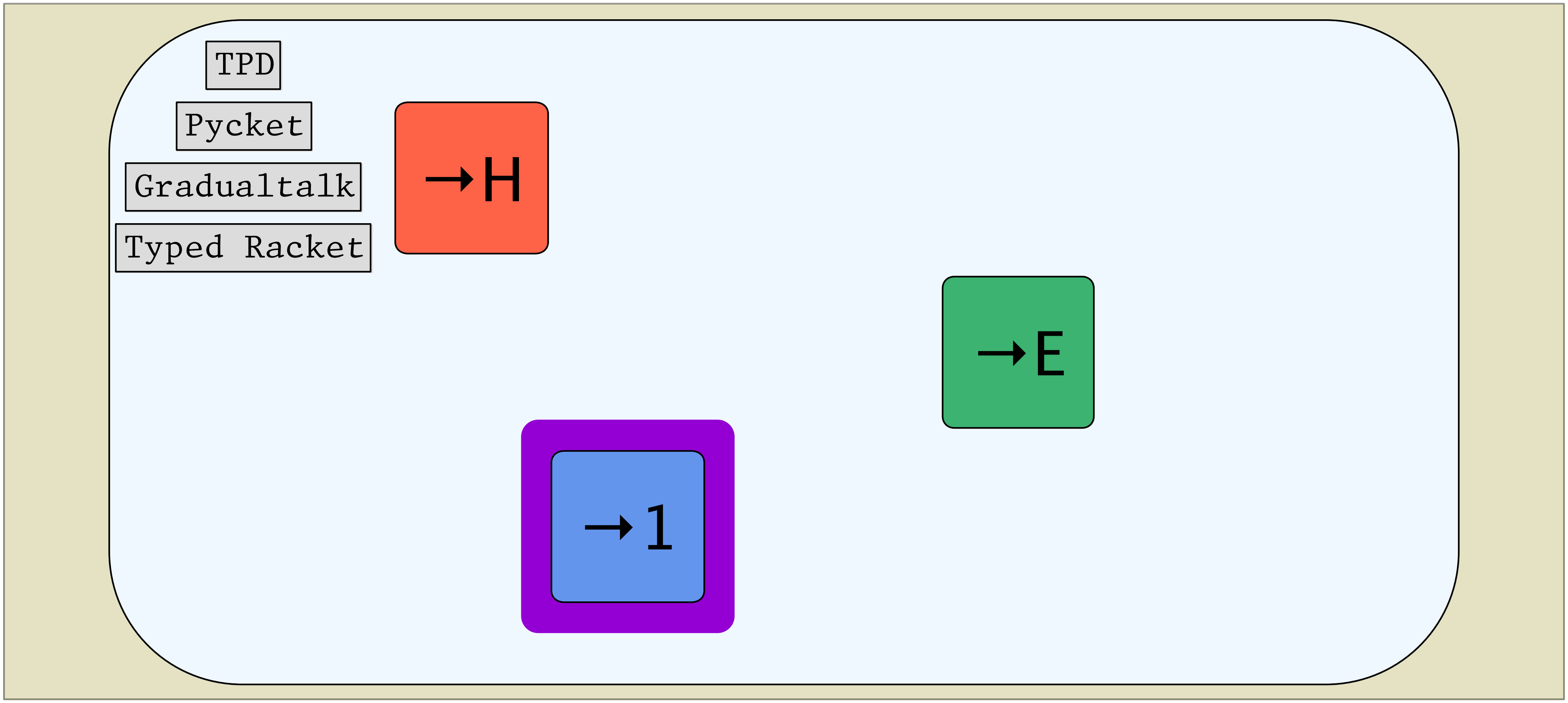




(the systems landscape)

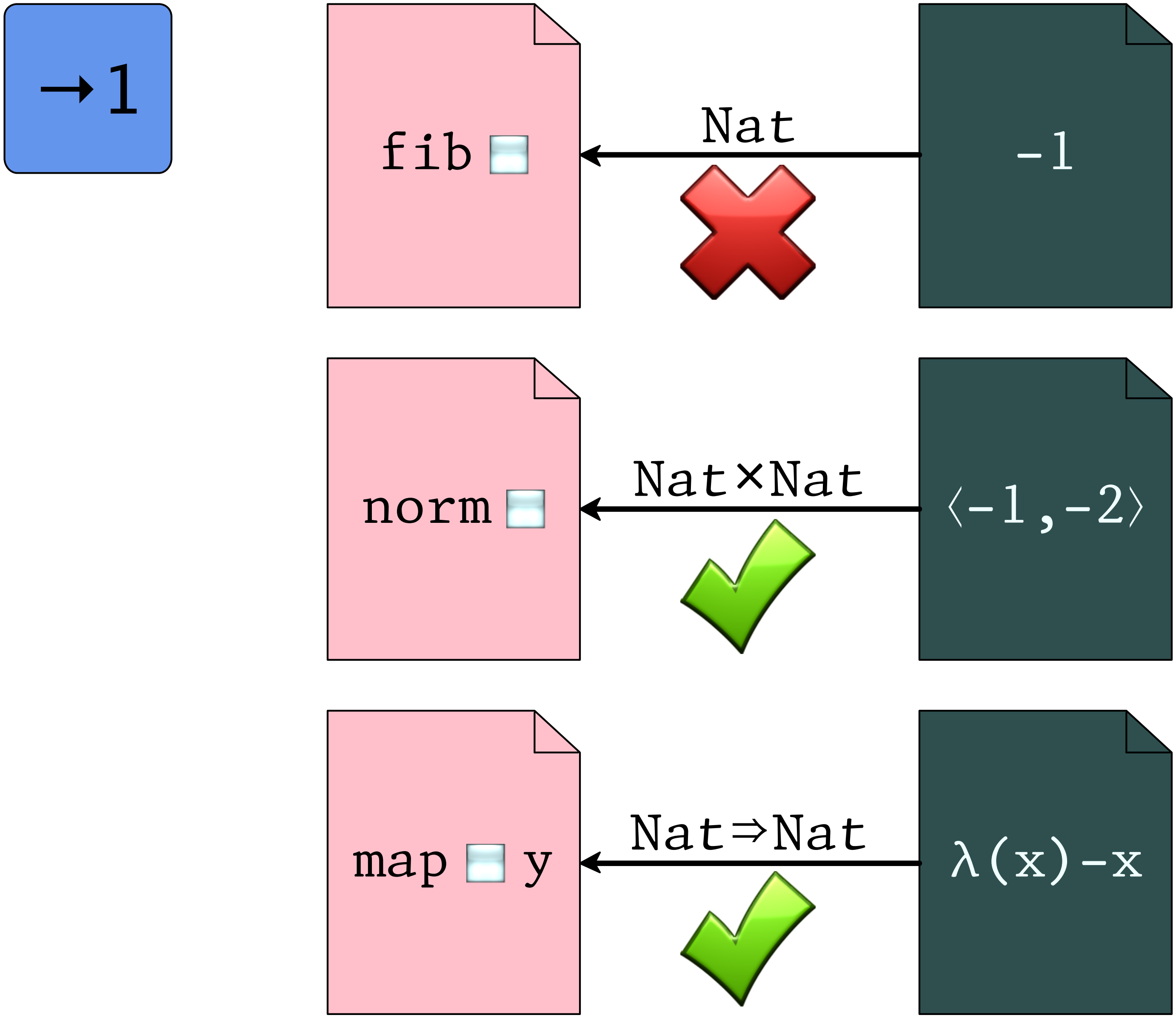


(the systems landscape)

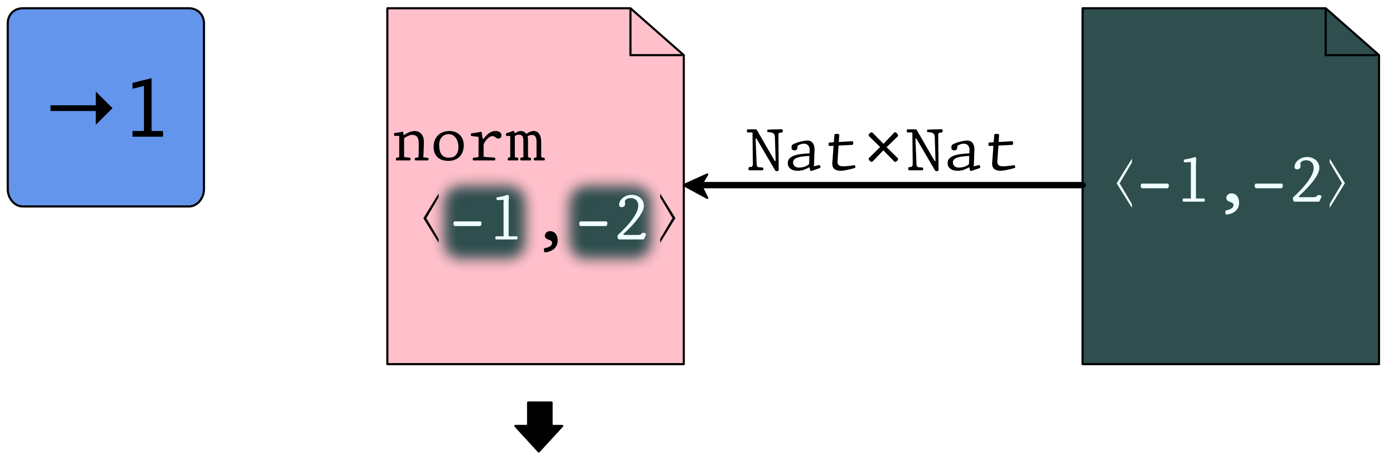


(the systems landscape)

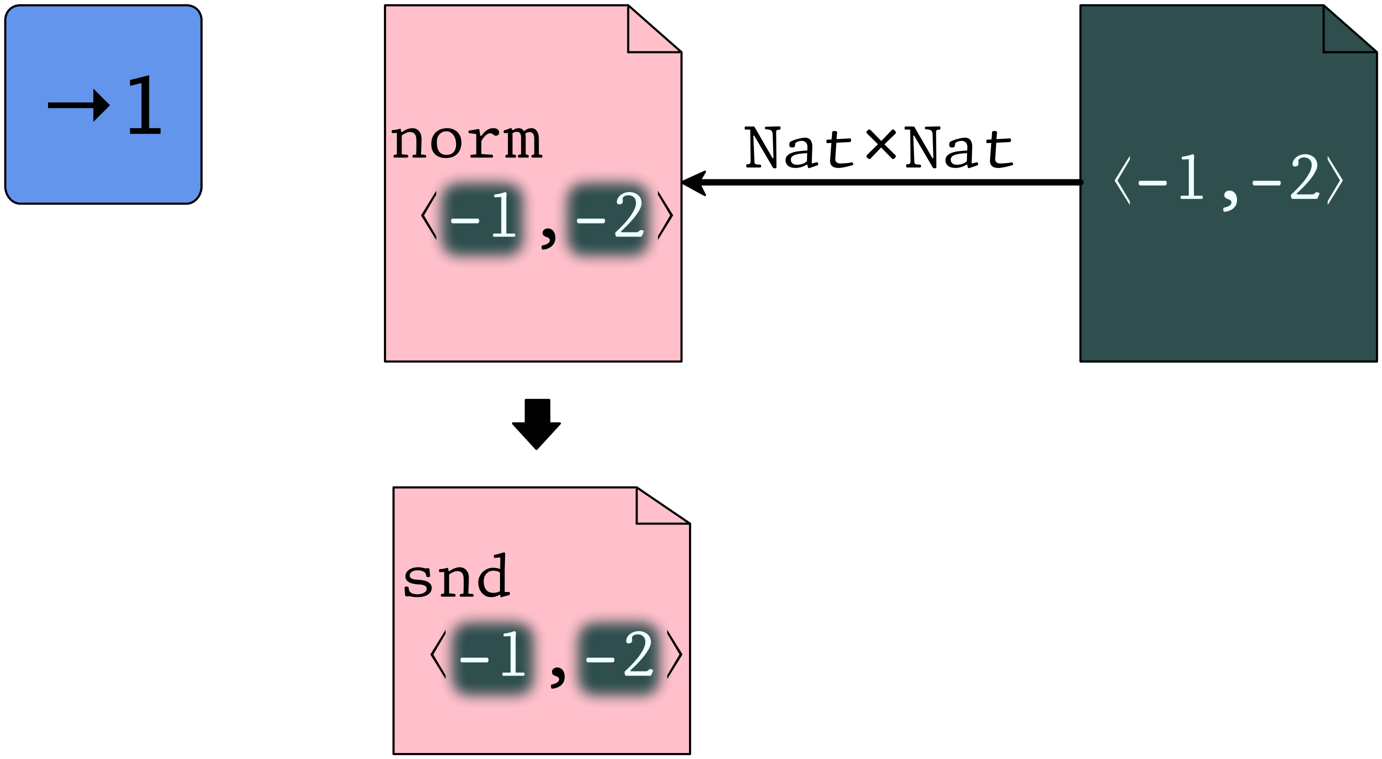
first-order (enforce type constructors)



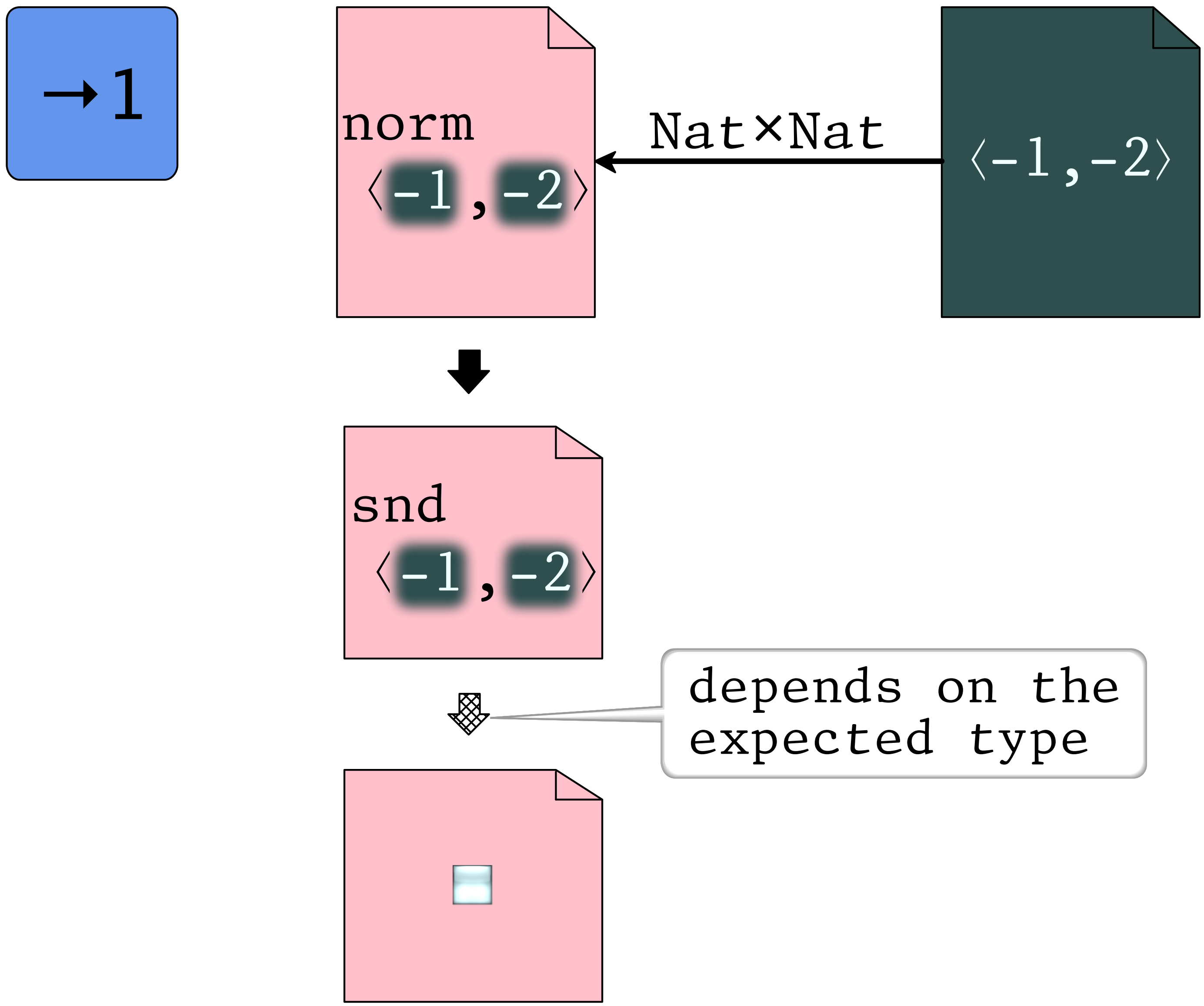
first-order (enforce type constructors)



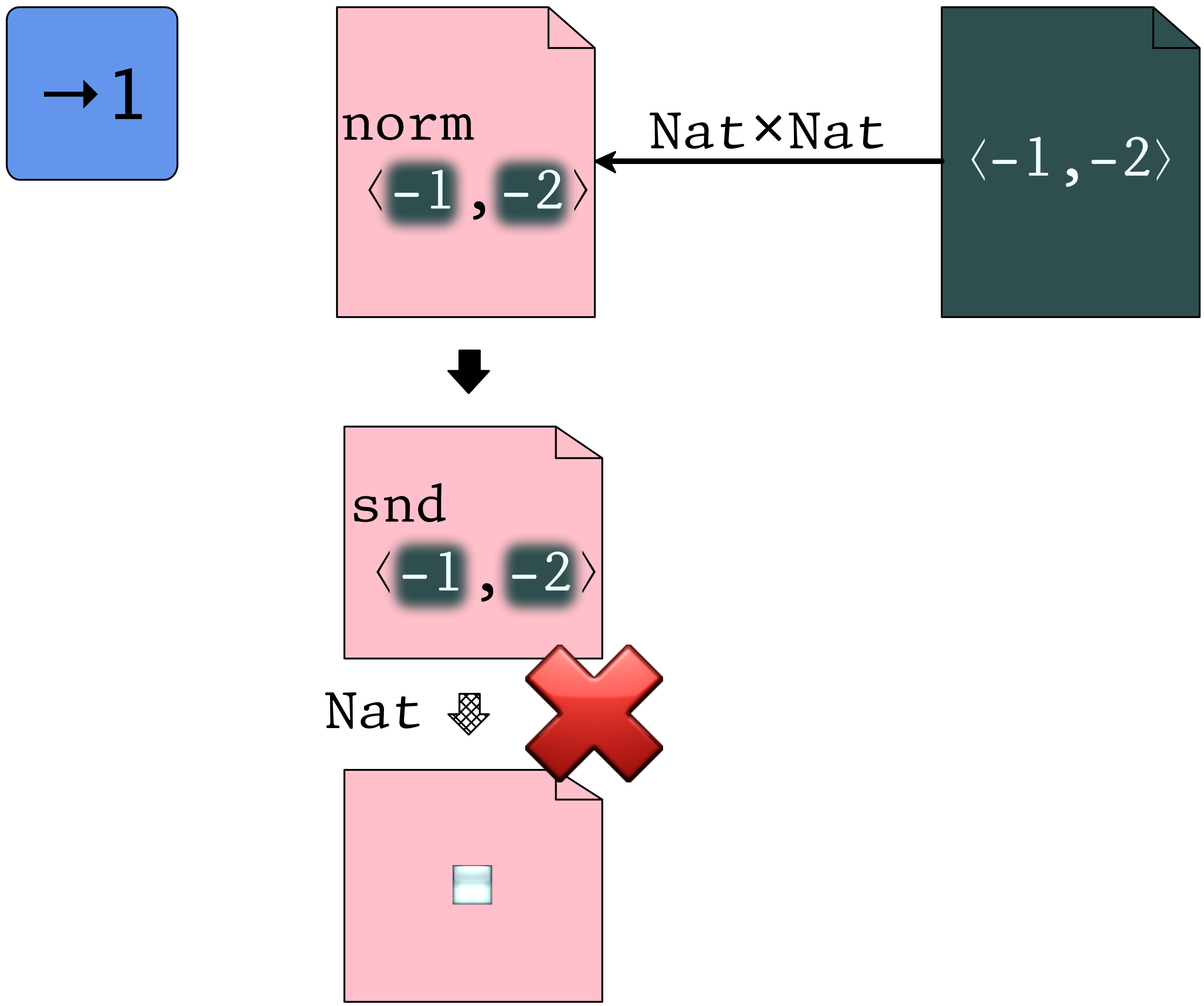
first-order (enforce type constructors)



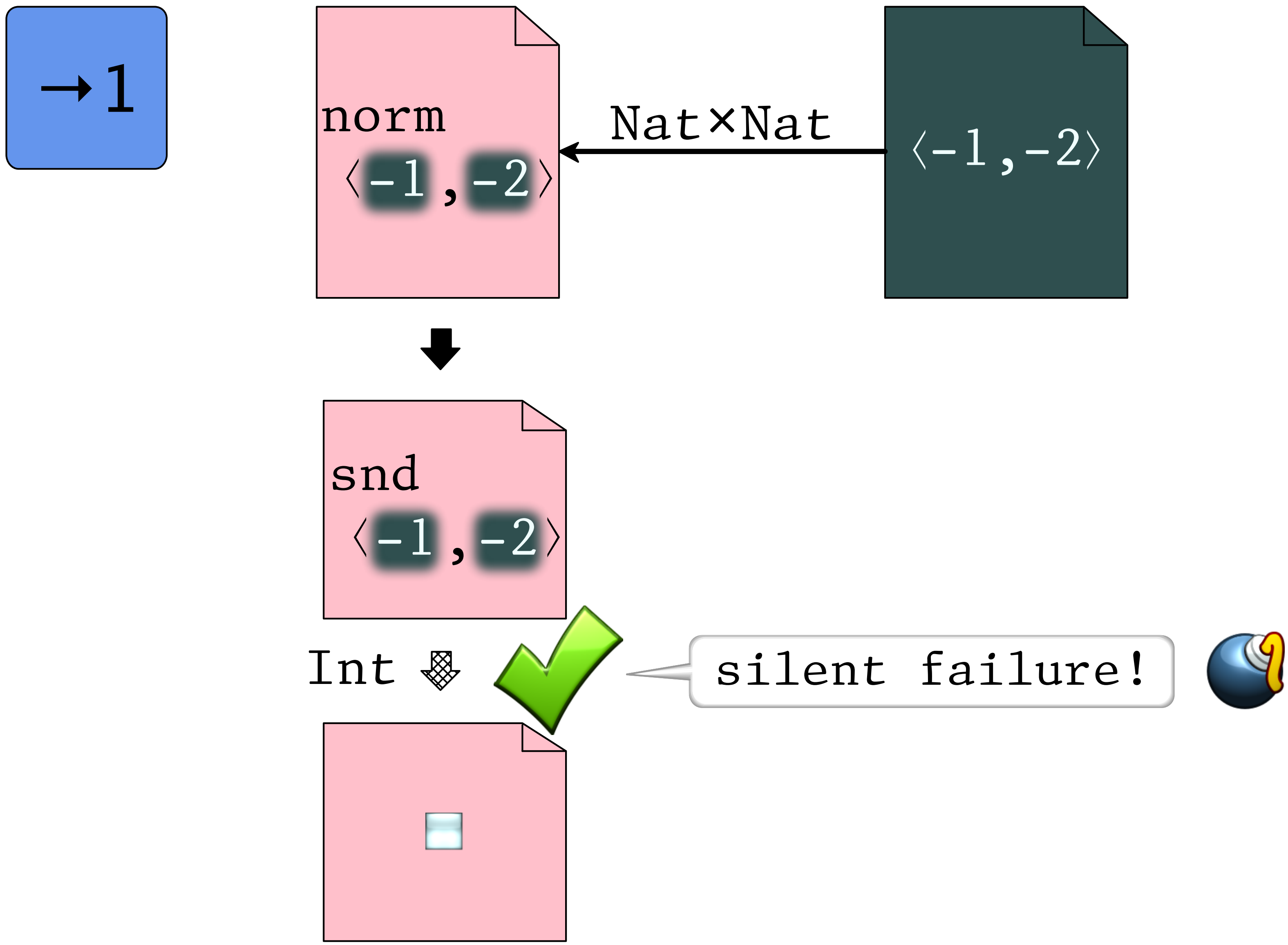
first-order (enforce type constructors)

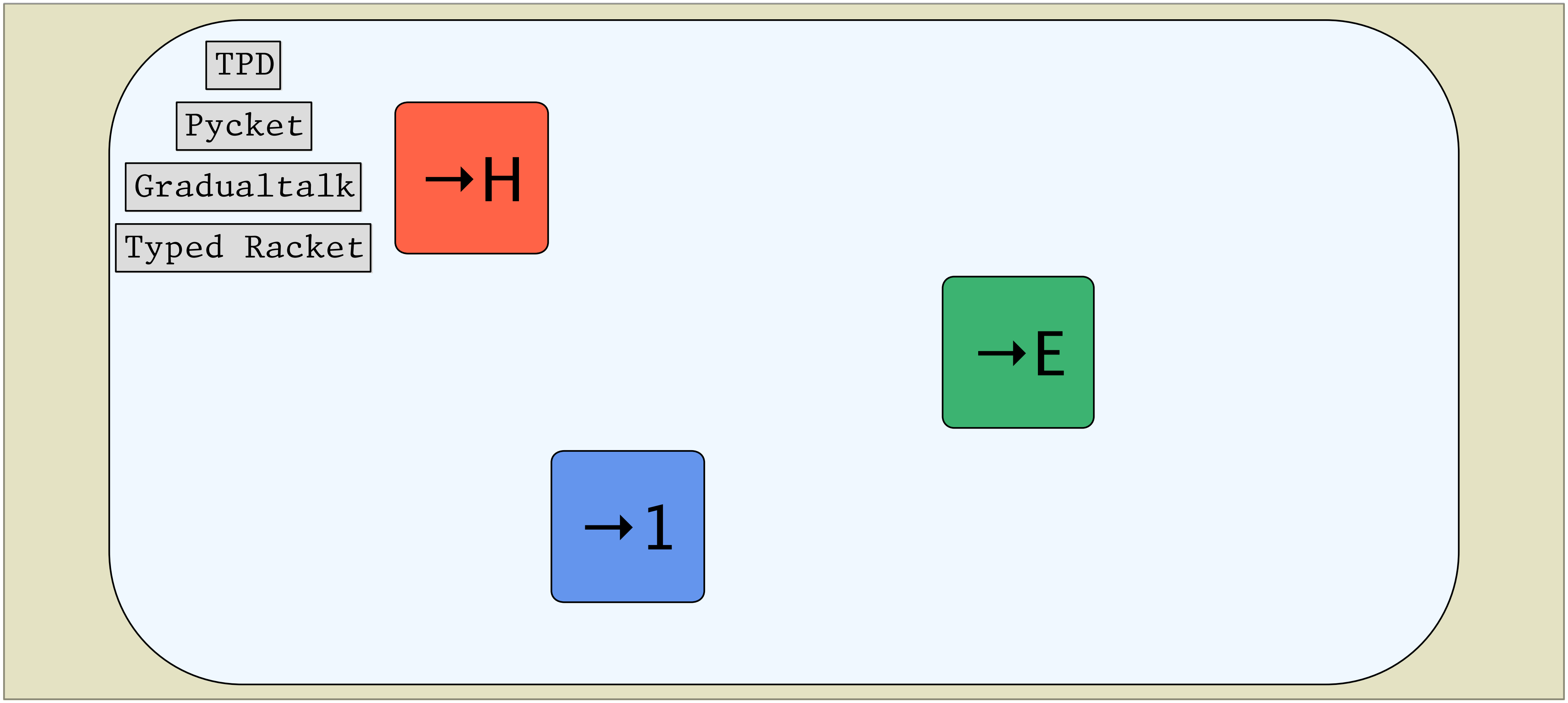


first-order (enforce type constructors)

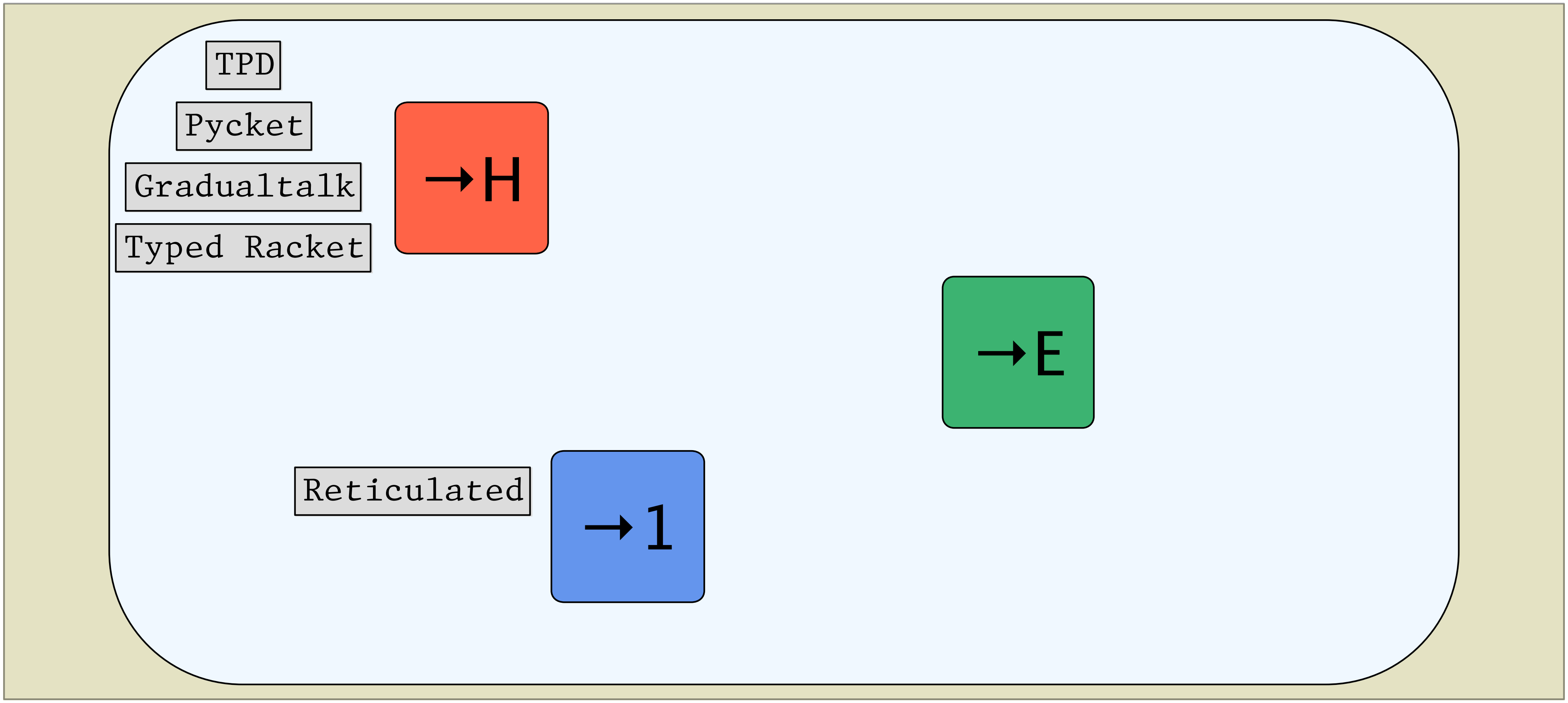


first-order (enforce type constructors)

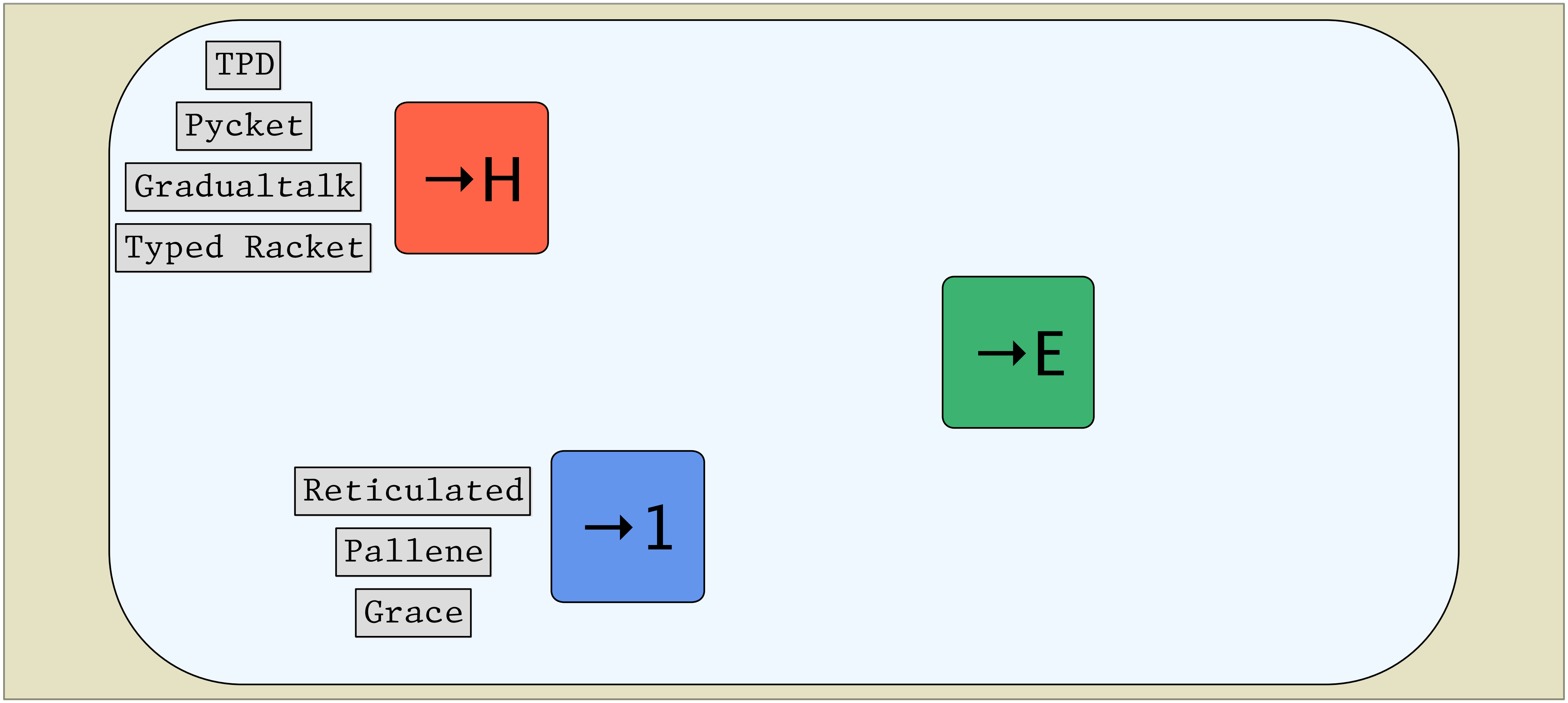




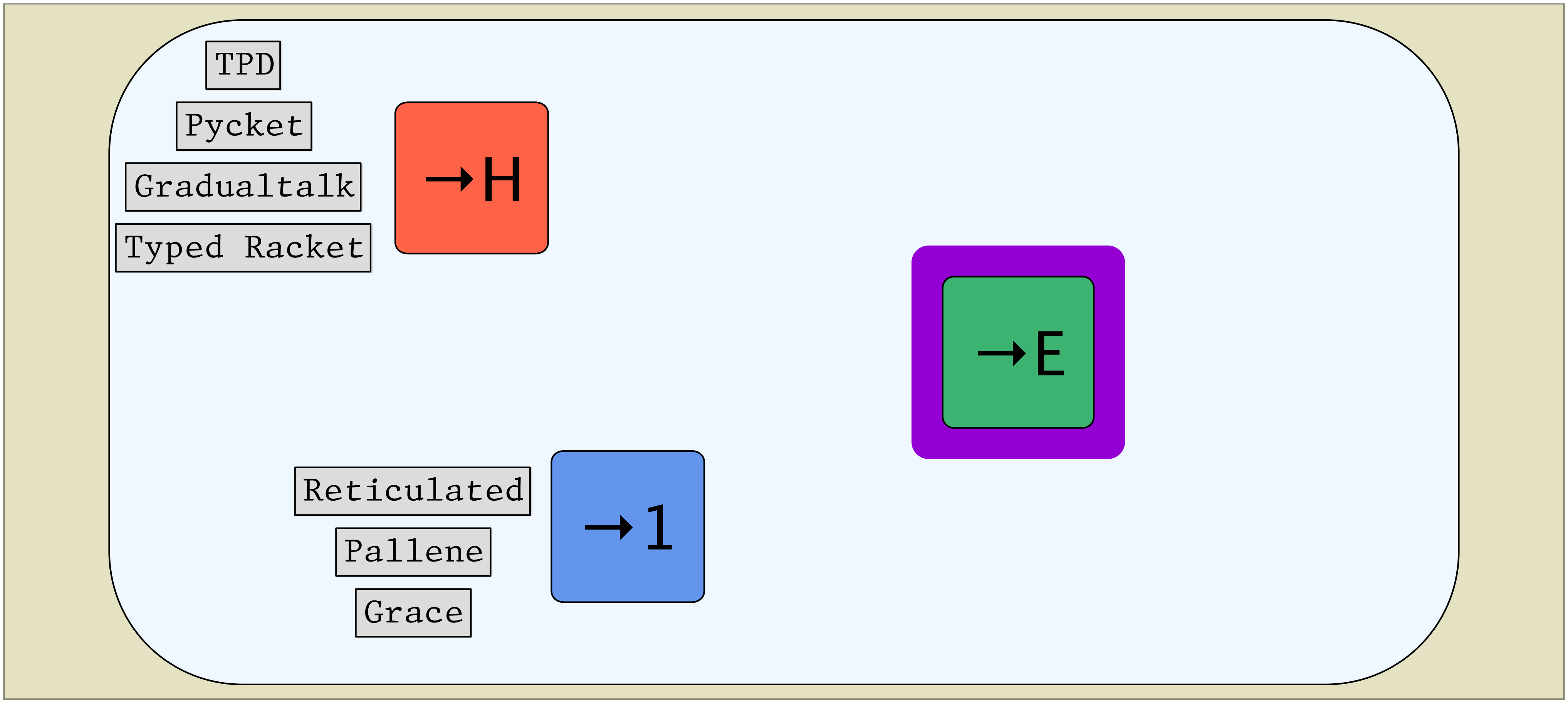
(the systems landscape)



(the systems landscape)

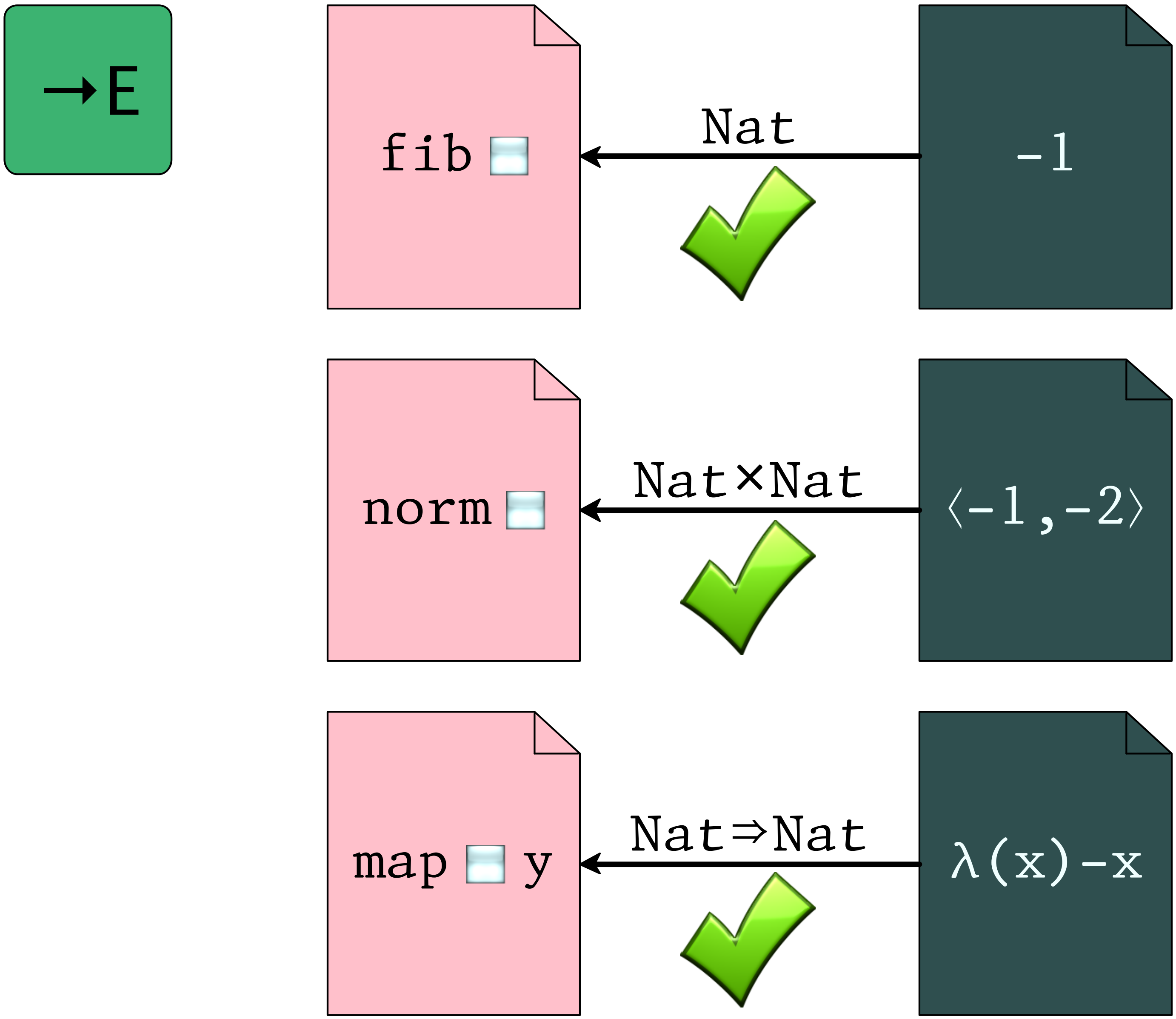


(the systems landscape)

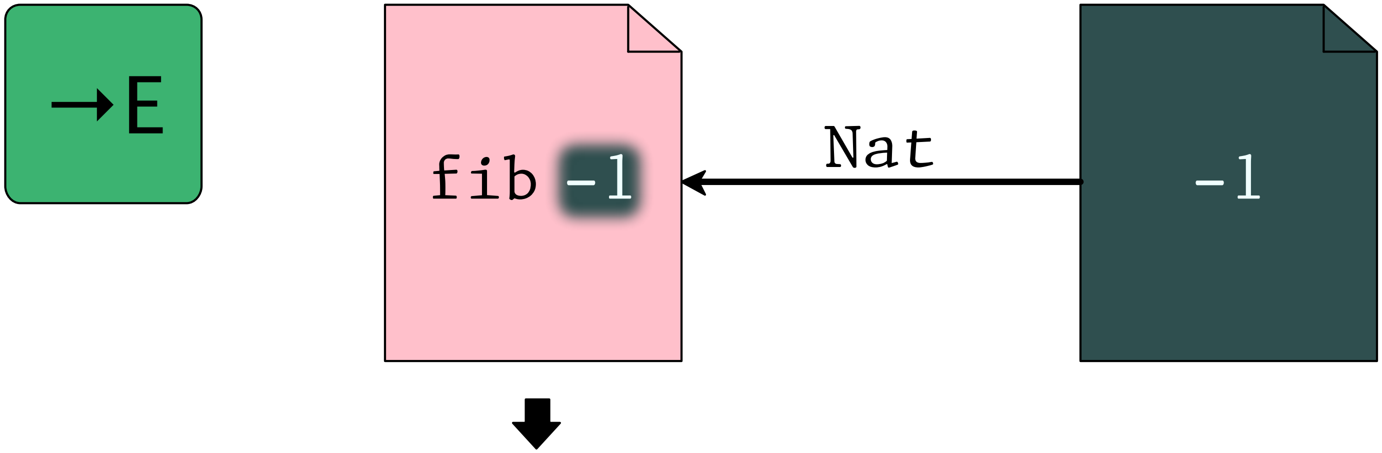


(the systems landscape)

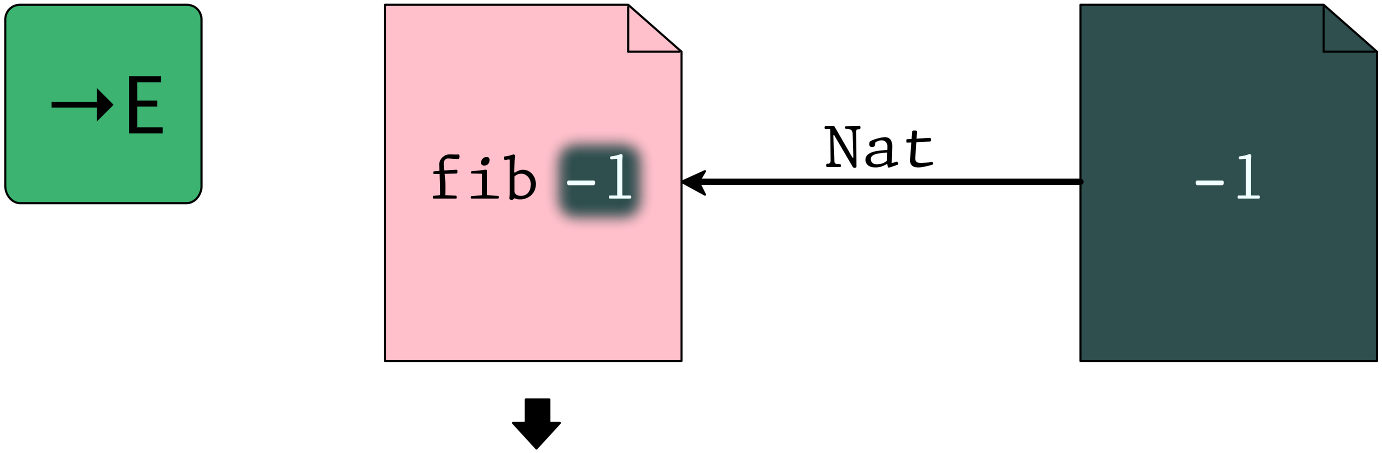
erasure (ignore types)



erasure (ignore types)



erasure (ignore types)

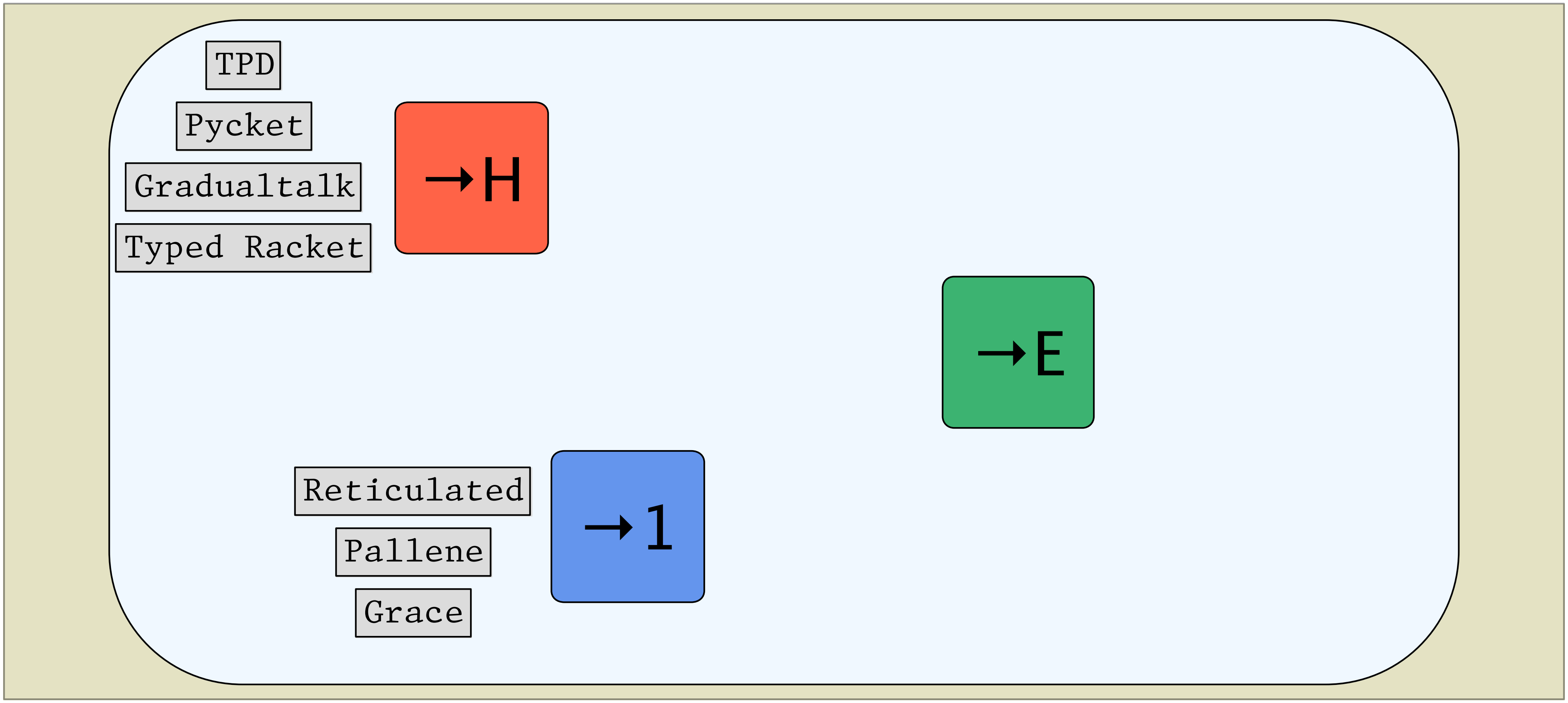


error?

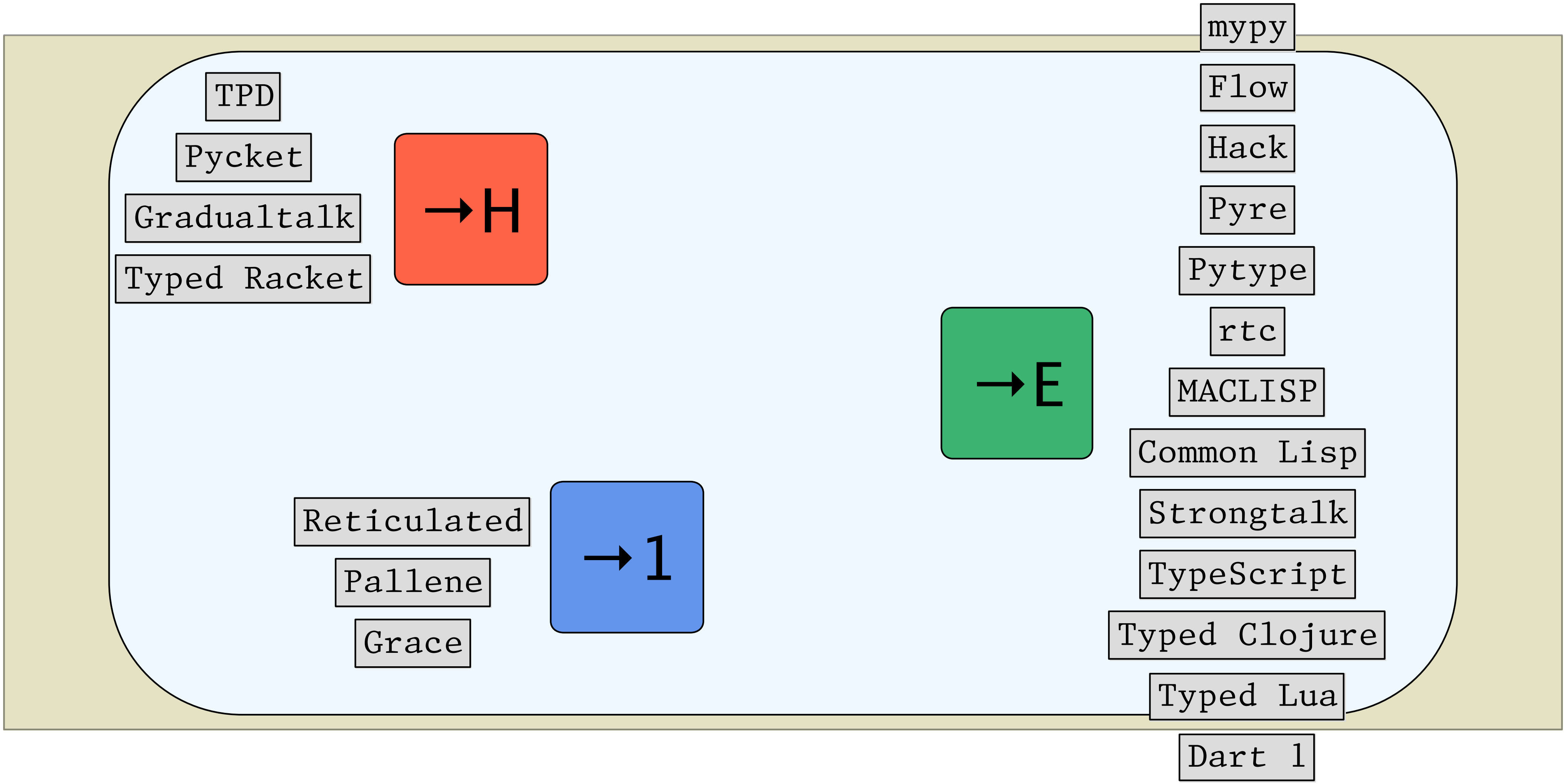
diverges?

0

???



(the systems landscape)

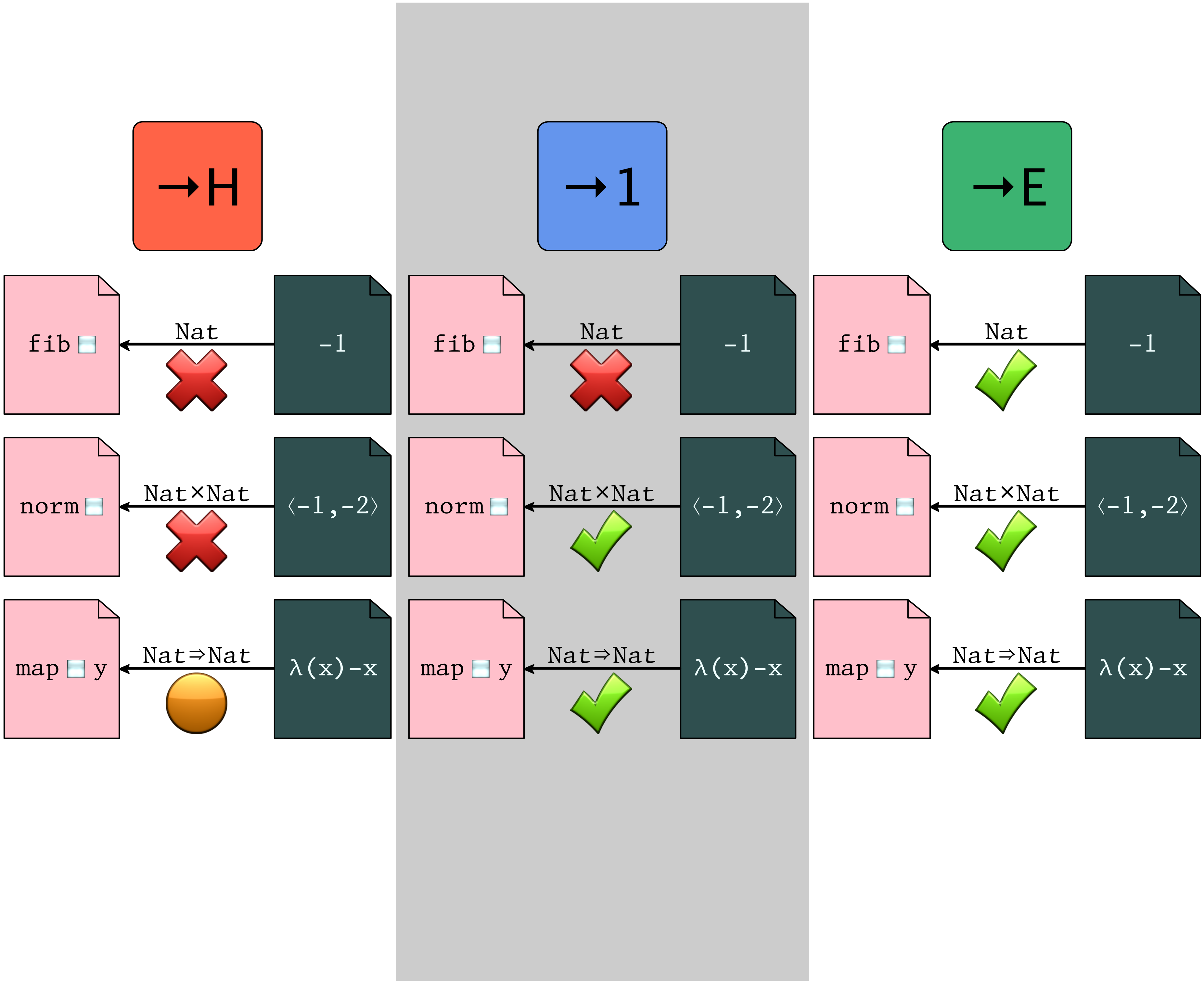


(the systems landscape)

→H

→1

→E



→H
→1
→E
fib
Nat
-1
norm
Nat×Nat
⟨-1,-2⟩
map y
Nat⇒Nat
λ(x)-x

All None

Type violations discovered


→H
⊃
→1
⊃
→E
All
None
Type violations discovered

Theorem (⊇):

- if e →1 Error
  then e →H Error

- if e →E Error
  then e →1 Error

Theorem (⊇):

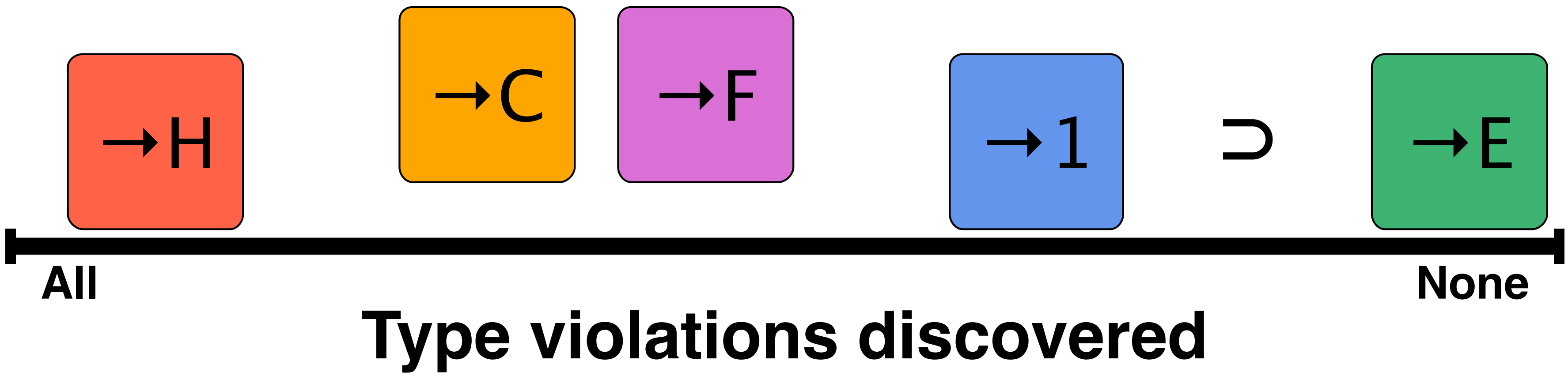
- if e →1 Error
  then e →H Error

- if e →E Error
  then e →1 Error

Counterexamples (⊉):

- see prev. slide

→H
⊃
→1
⊃
→E
All
None
Type violations discovered

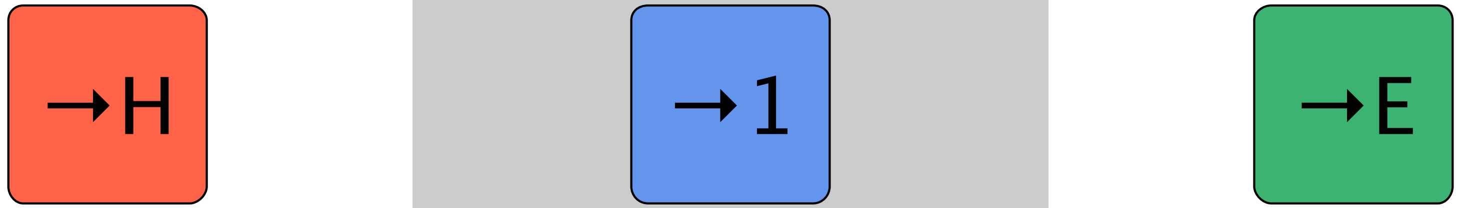


Appendix: two other semantics

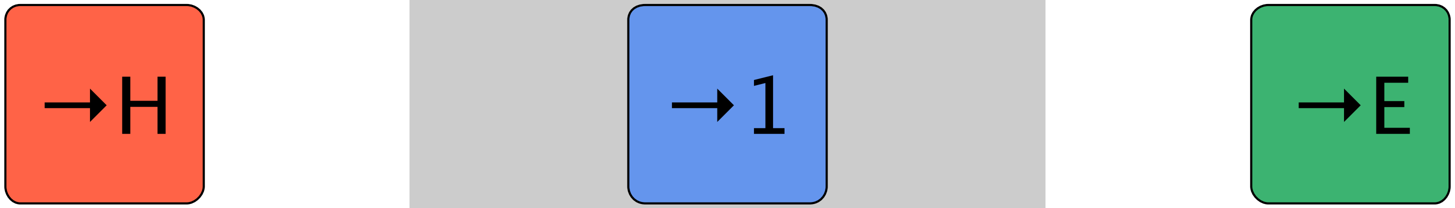
→H

→1

→E



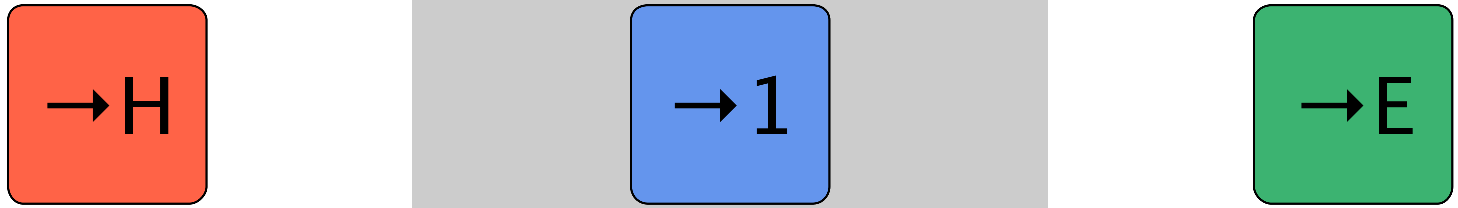

```
Type Soundness (simplified):

 if ⊢e:τ then either:

 - e ->* v and ⊢v:τ

 - e diverges

 - e ->* Error
```



Type Soundness (simplified):

if $\vdash e : \tau$ then either:

- e ->* v and $\vdash v : \tau$
- e diverges
- e ->* Error



Type Soundness (simplified):

if $\vdash e : \tau$ then either:

- e ->* v and $\vdash v : \tau$
- e diverges
- e ->* Error

$\rightarrow_H$ $\rightarrow_1$ $\rightarrow_E$

Type Soundness (simplified):

if $\vdash e : \tau$ then either:

- e ->* v and $\vdash v : \tau$
- e diverges
- e ->* Error

→H

→1

→E

→H Soundness:

if ⊢e:τ then either:

- e ->* v and ⊢v:τ
- e diverges
- e ->* Error

→H

→1

→E

→H Soundness:

if $\vdash e : \tau$ then either:

- e ->* v and $\vdash v : \tau$
- e diverges
- e ->* Error

→1 Soundness:

if $\vdash e : \tau$ then either:

- e ->* v and $\vdash v : C(\tau)$
- e diverges
- e ->* Error

→H

→H Soundness:

if ⊢e:τ then either:

- e ->* v and ⊢v:τ
- e diverges
- e ->* Error

→1

→1 Soundness:

if ⊢e:τ then either:

- e ->* v and ⊢v:C(τ)
- e diverges
- e ->* Error

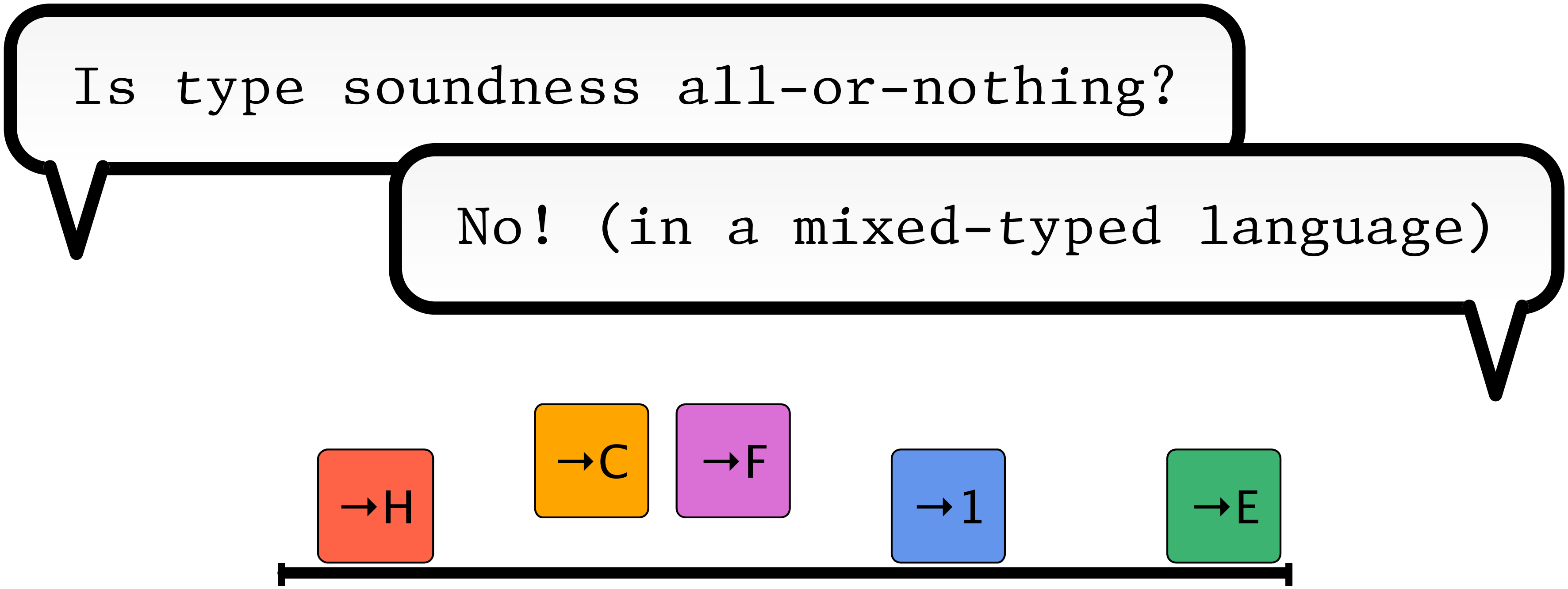
→E

→E Soundness:

if ⊢e:τ then either:

- e ->* v and ⊢v
- e diverges
- e ->* Error

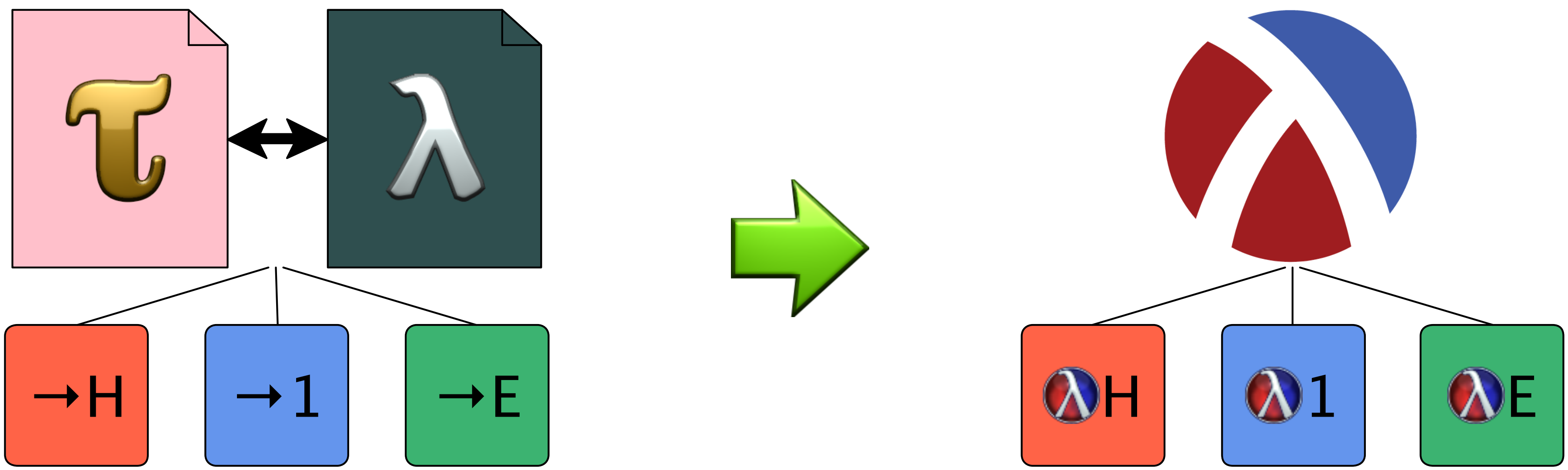
Is type soundness all-or-nothing?



Is type soundness all-or-nothing?
No! (in a mixed-typed language)
→H
→C
→F
→1
→E

Implementation

How does type soundness affect performance?



`model => implementation`

3 Compilers

3 Compilers

H

1

E

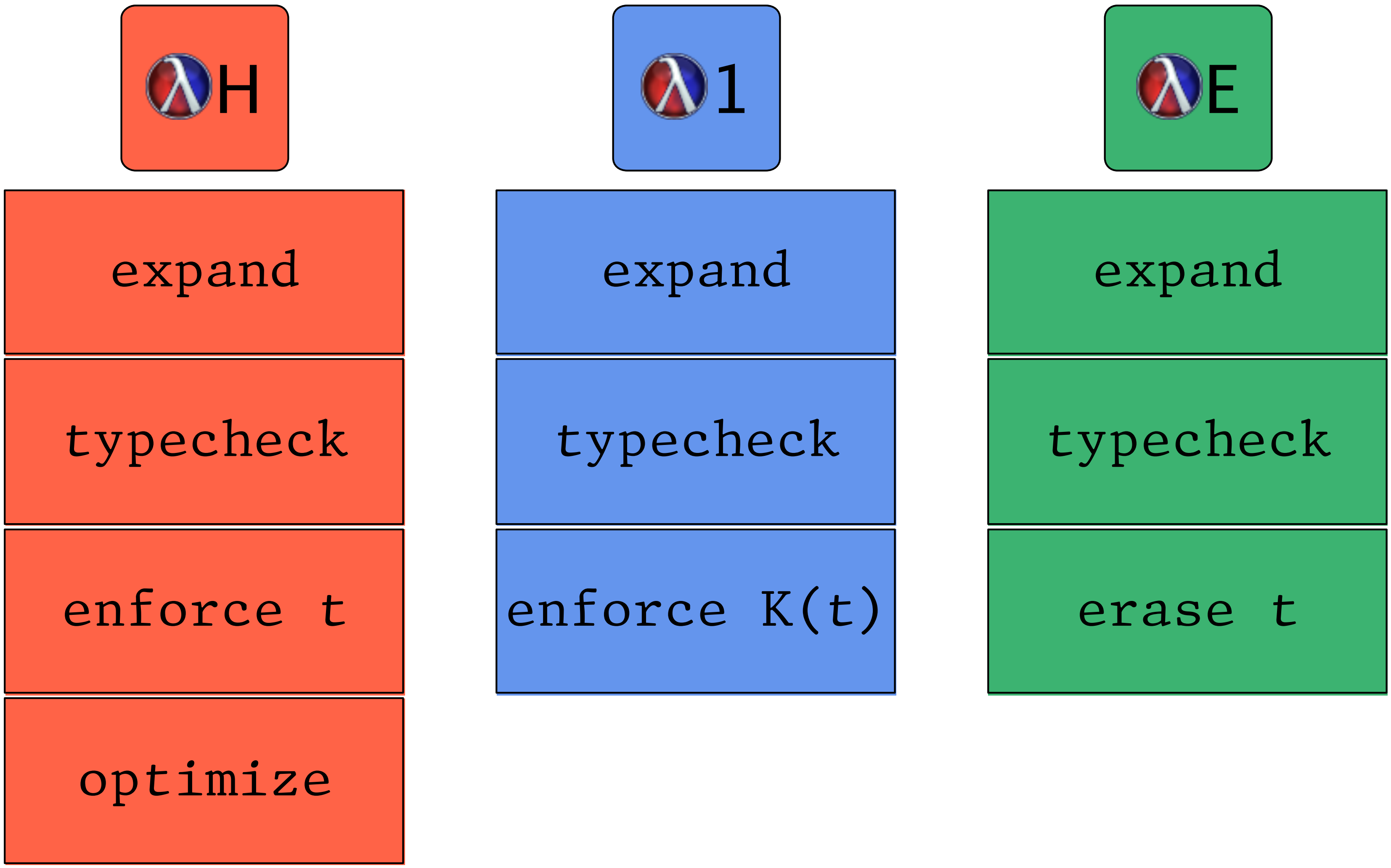
expand

typecheck

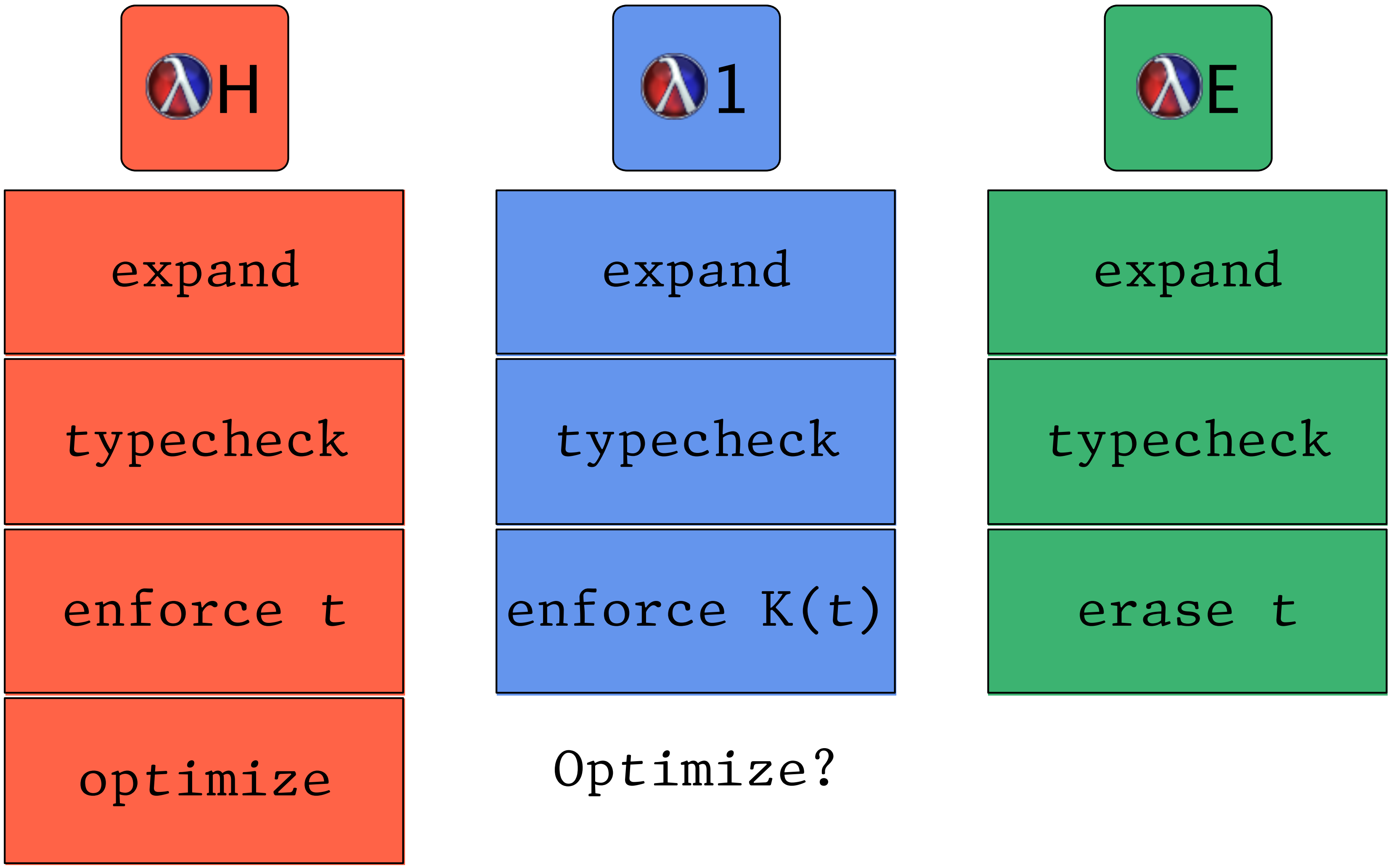
enforce t

optimize

3 Compilers


H
1
E
expand
typecheck
enforce t
optimize
expand
typecheck
enforce K(t)
expand
typecheck
erase t

3 Compilers

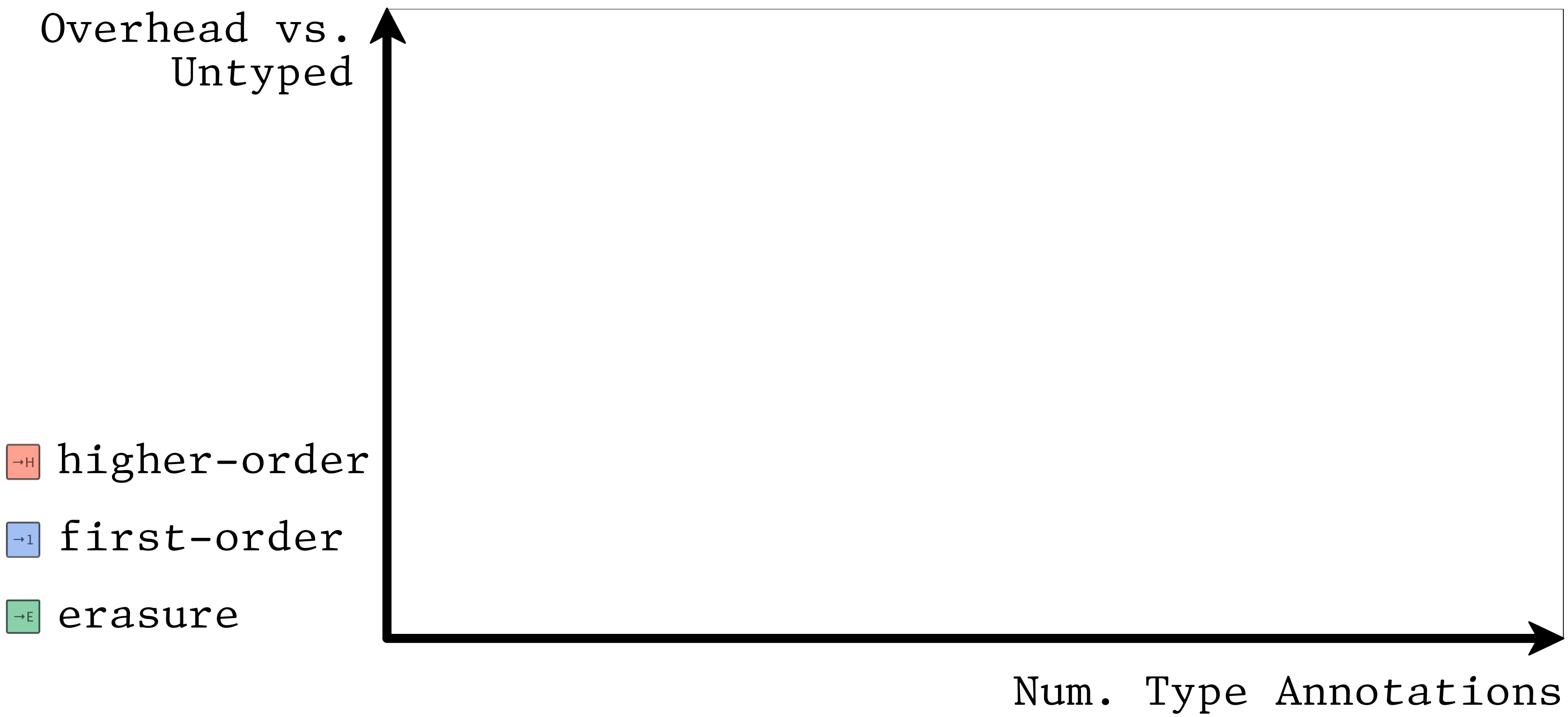

H
1
E
expand
typecheck
enforce t
optimize
expand
typecheck
enforce K(t)
Optimize?
expand
typecheck
erase t

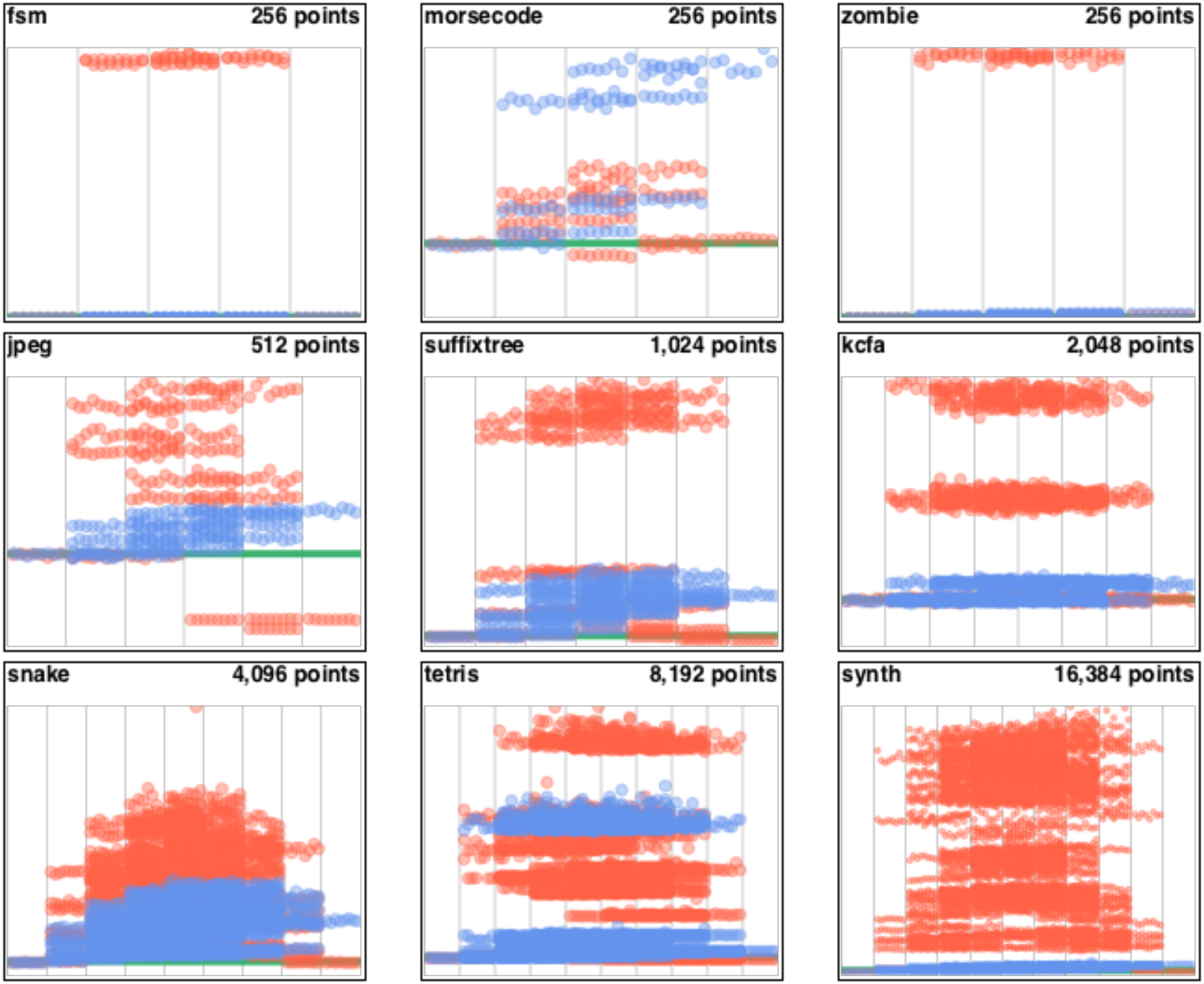
Experiment (method from POPL'16)

- 10 benchmark programs
- 2 to 10 modules each
- 4 to 1024 configurations each
- compare overhead to untyped

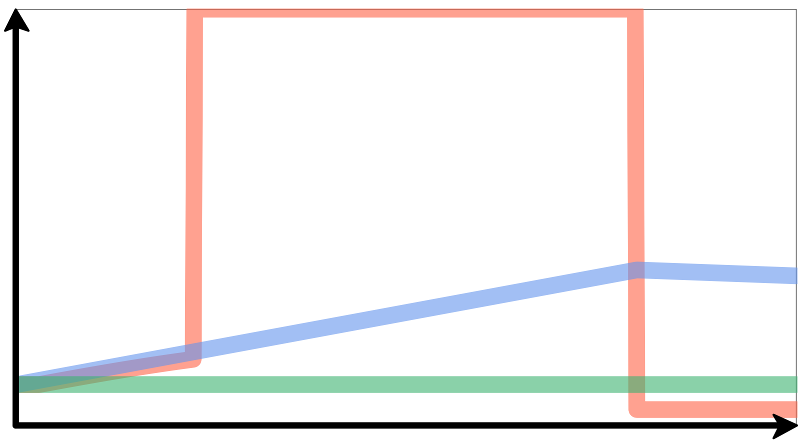
docs.racket-lang.org/gtp-benchmarks

Soundness vs. Performance

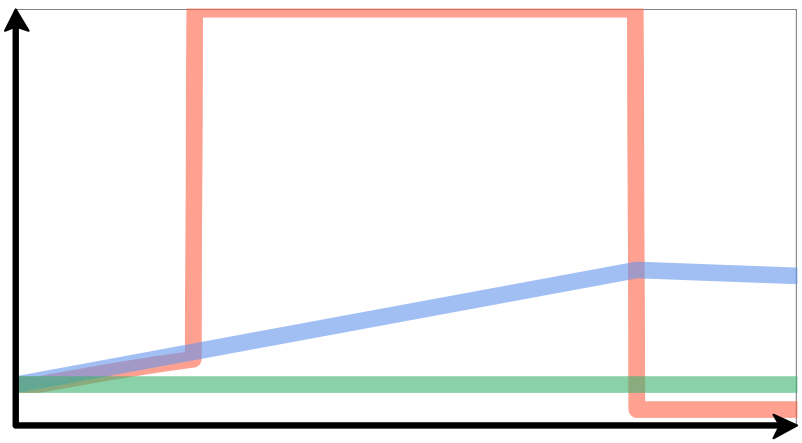


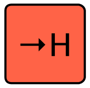

fsm
256 points
morsecode
256 points
zombie
256 points
jpeg
512 points
suffixtree
1,024 points
kcfa
2,048 points
snake
4,096 points
tetris
8,192 points
synth
16,384 points

Performance Implications



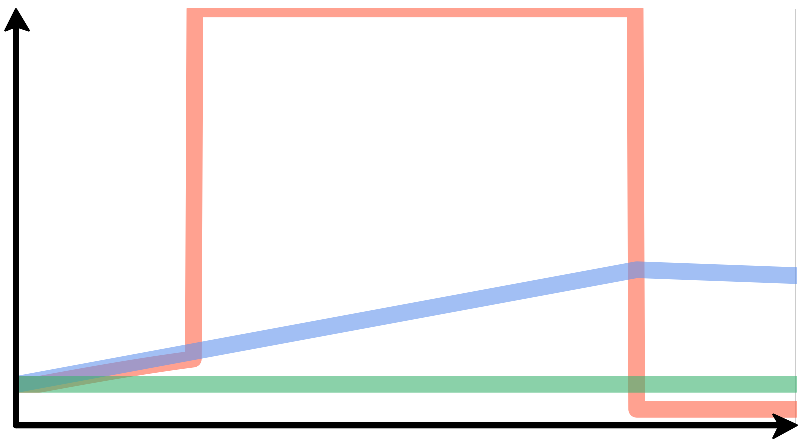
Performance Implications





add types to remove all critical boundaries

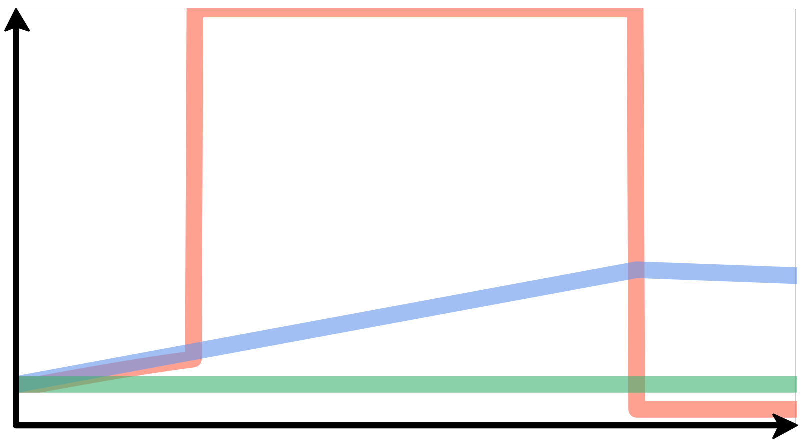
Performance Implications



→H add types to remove all critical boundaries

→1 add types sparingly

Performance Implications



→H add types to remove all critical boundaries

→1 add types sparingly

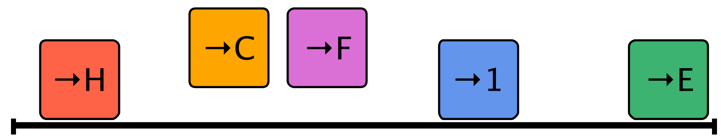
→E add types anywhere, doesn't matter

Takeaways

Takeaways

Theorists:

`type soundness is NOT all-or-nothing`



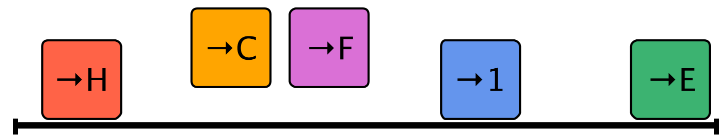
Takeaways

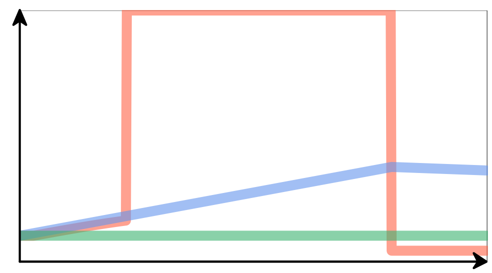
Theorists:

`type soundness is NOT all-or-nothing`

Implementors:

`can we change the performance landscape?`

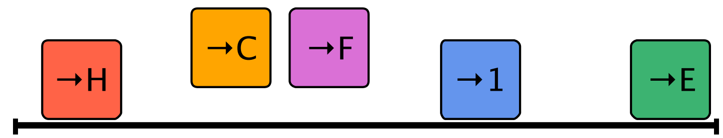




Takeaways

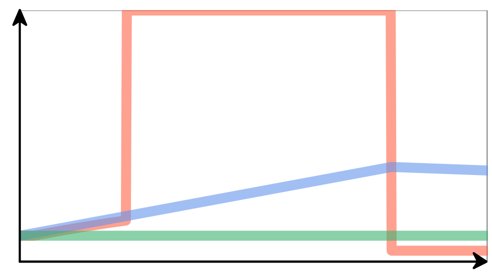
Theorists:

type soundness is NOT all-or-nothing



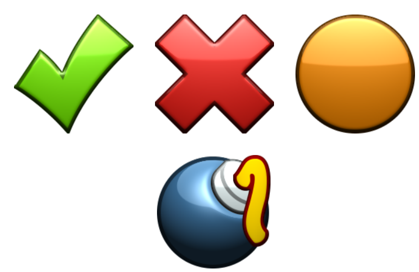
Implementors:

can we change the performance landscape?



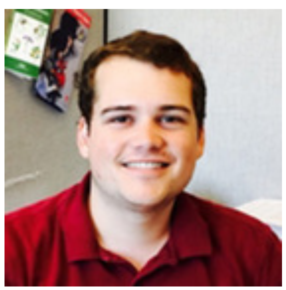
Users:

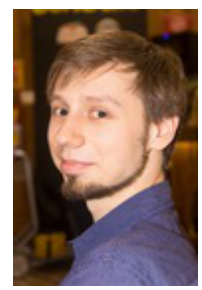
soundness affects **run**-time and **debug**-time



Special Thanks



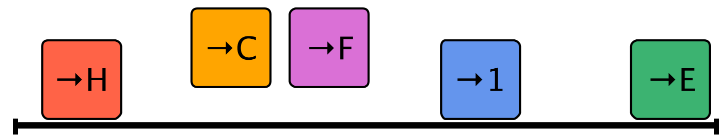



Takeaways

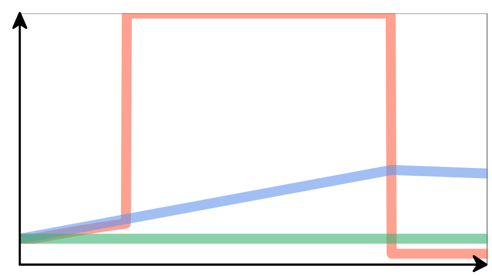
Theorists:

type soundness is NOT all-or-nothing



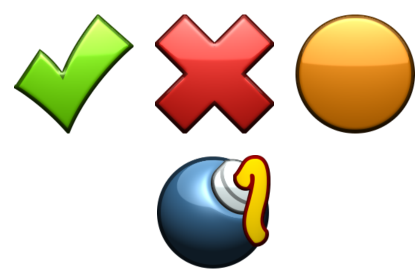
Implementors:

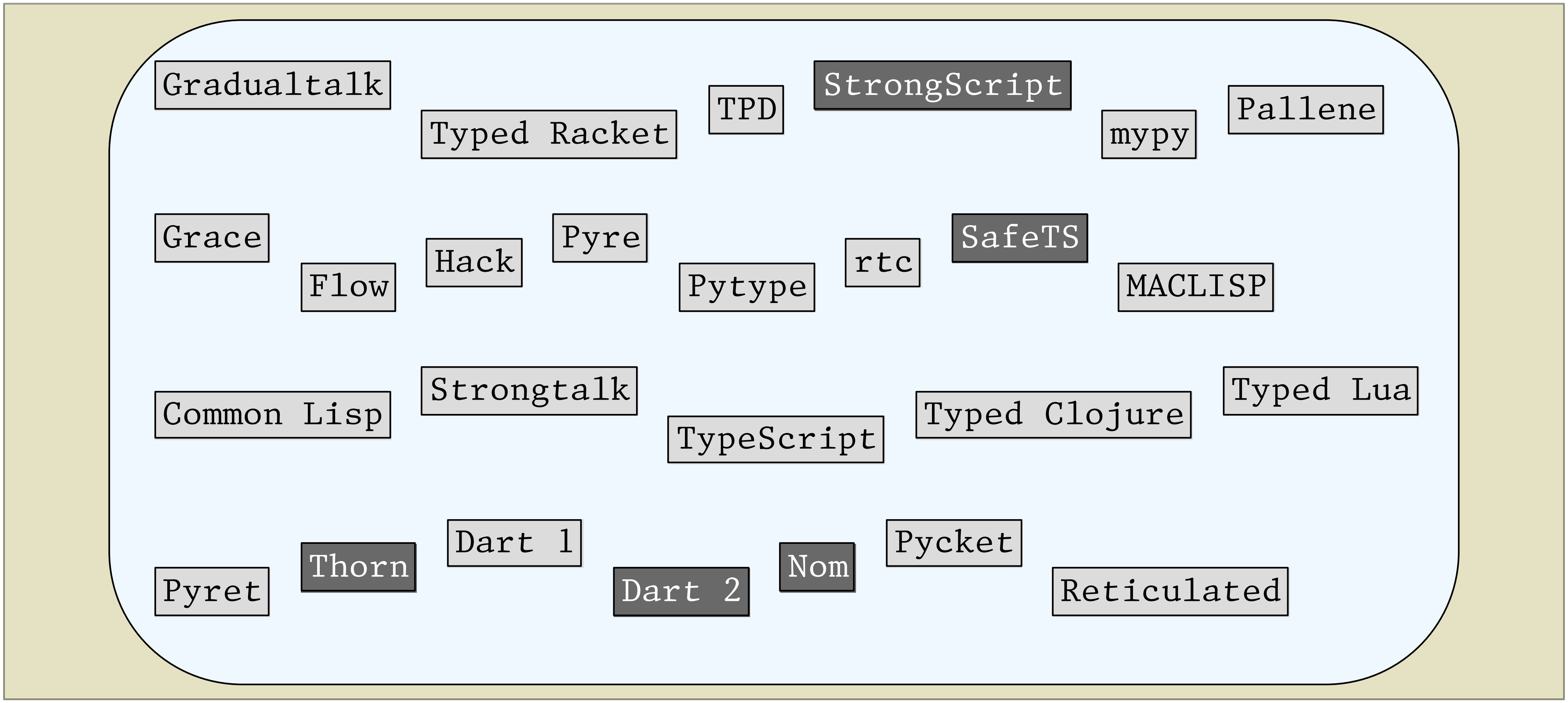
can we change the performance landscape?



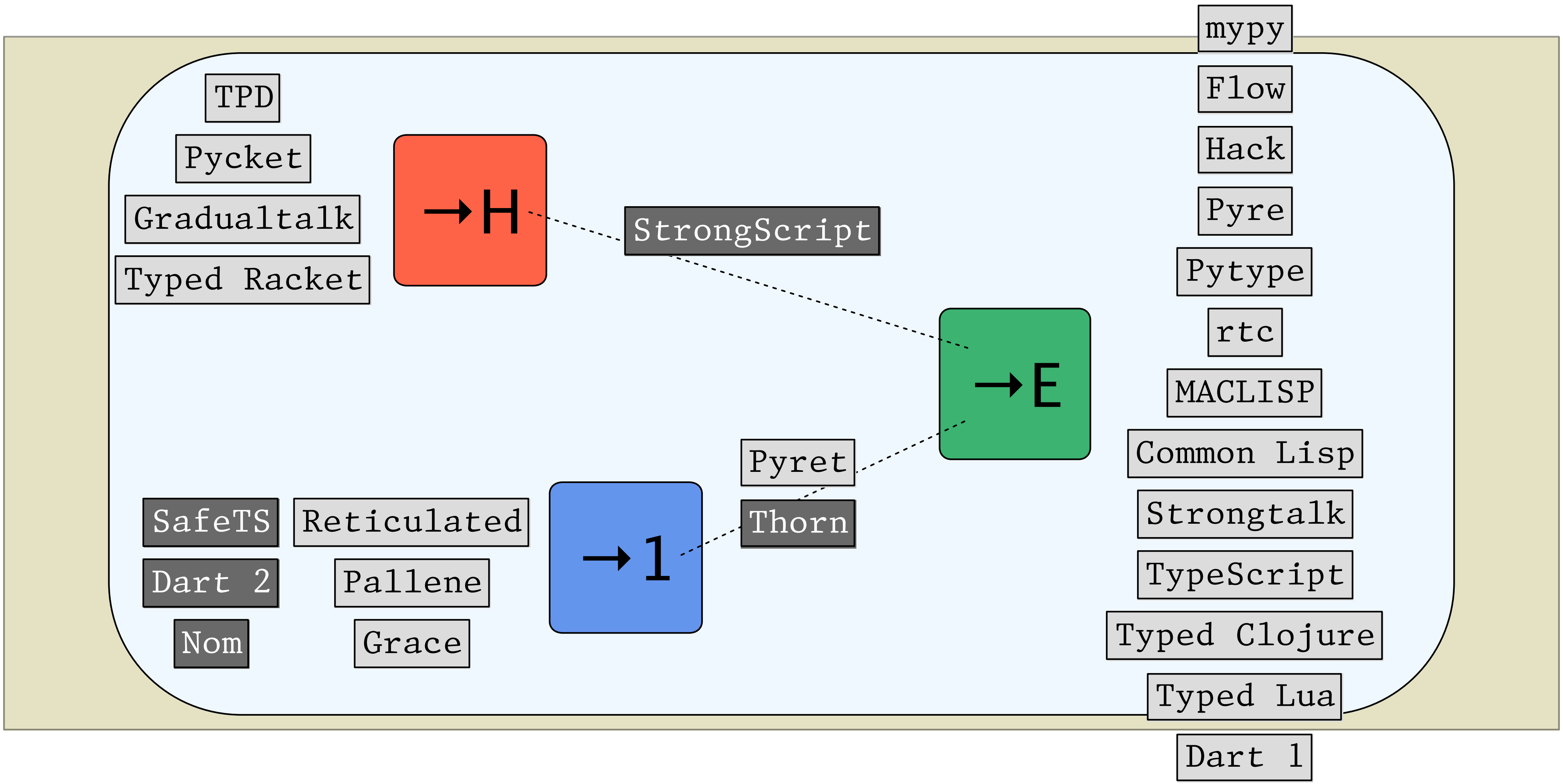
Users:

soundness affects **run**-time and **debug**-time



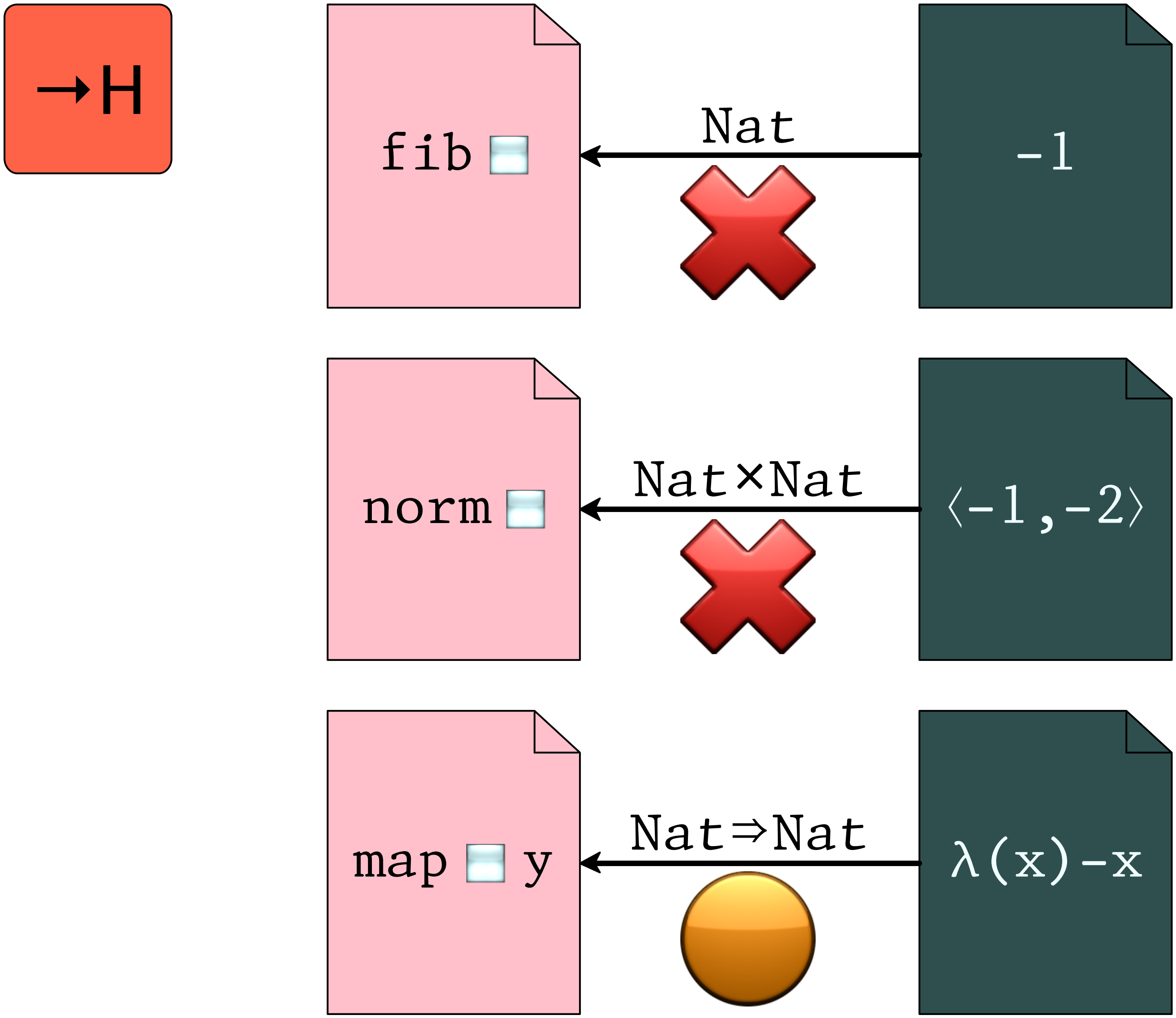


(the systems landscape)

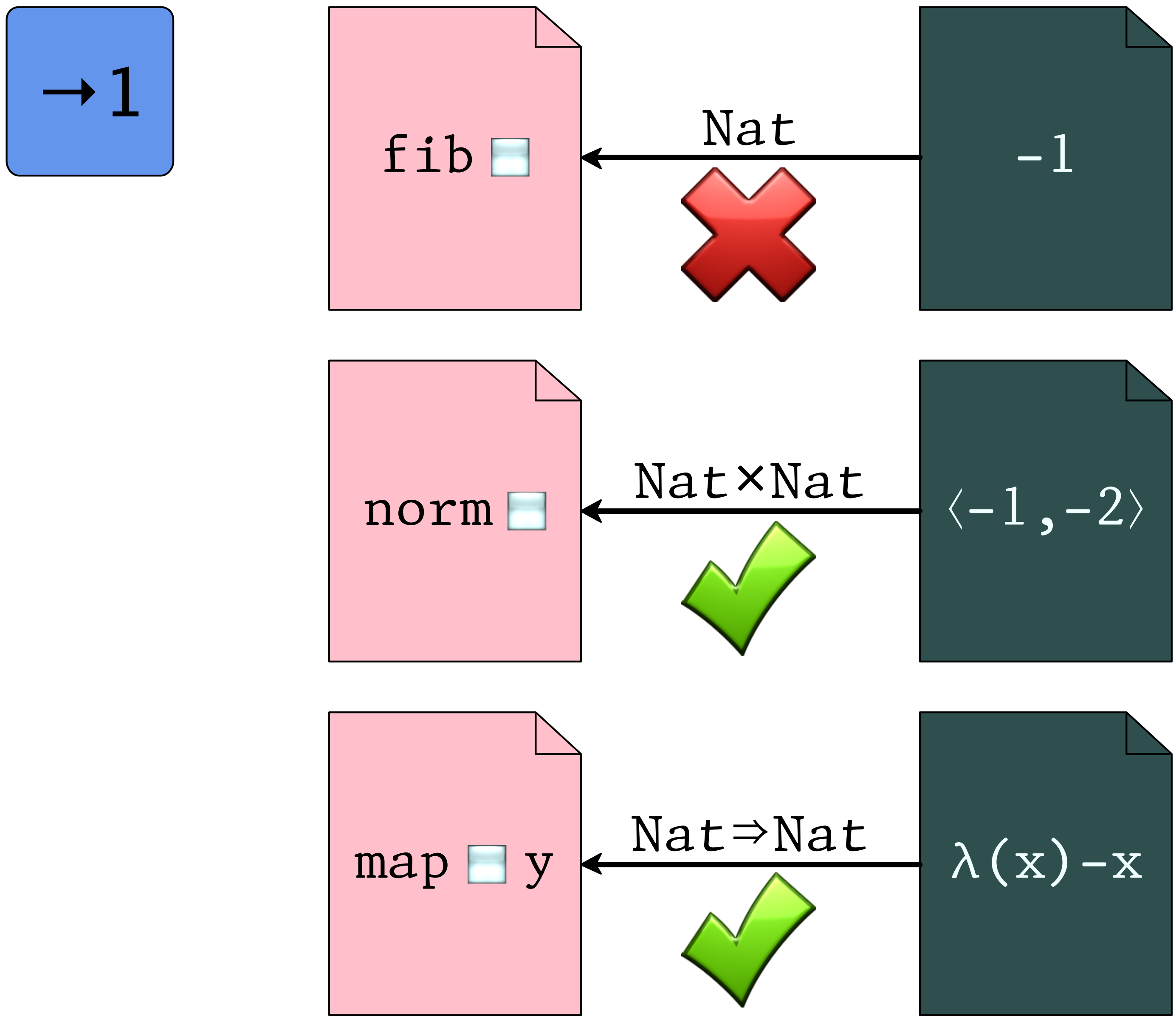


(the systems landscape)

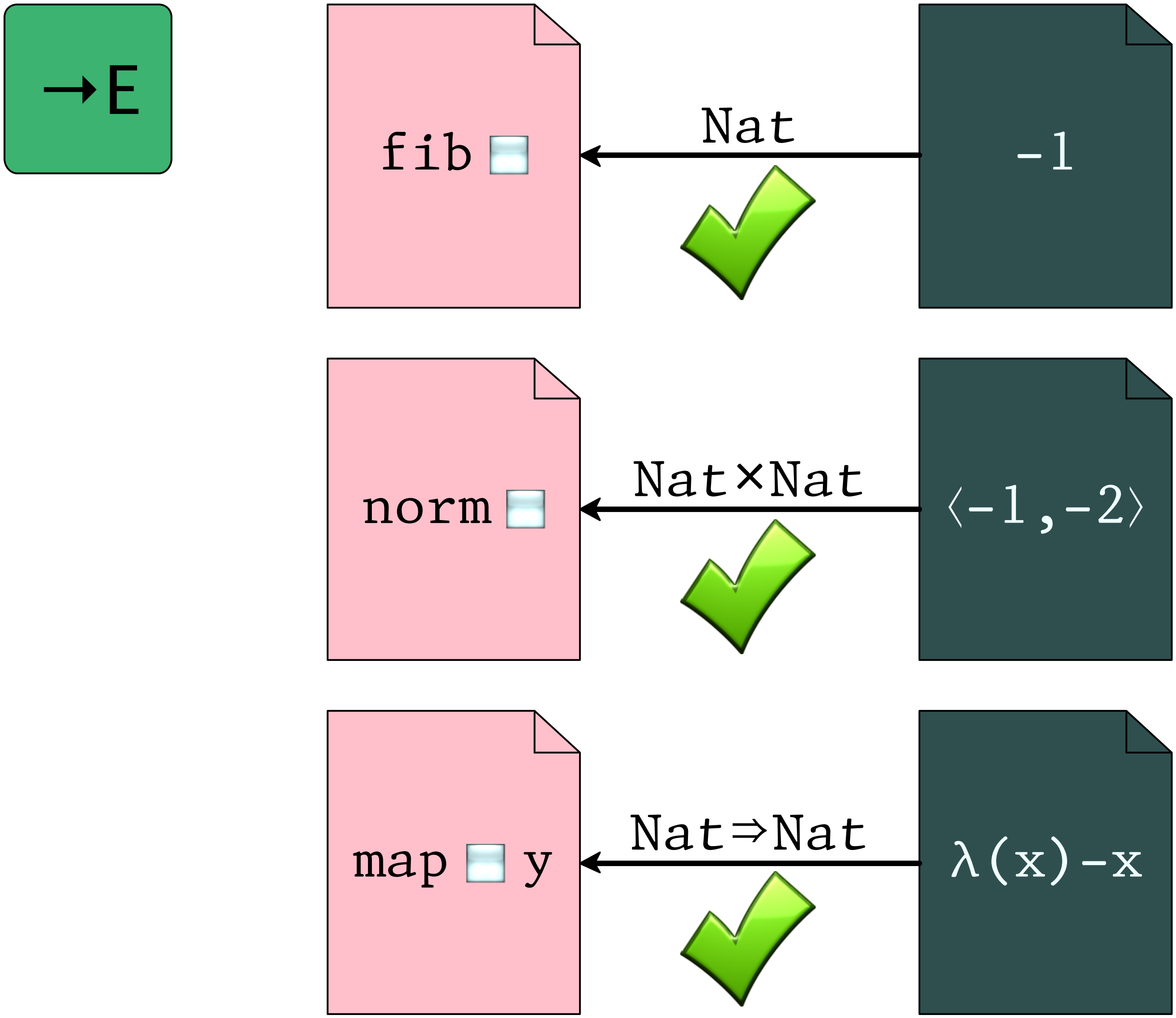
higher-order (enforce full types)

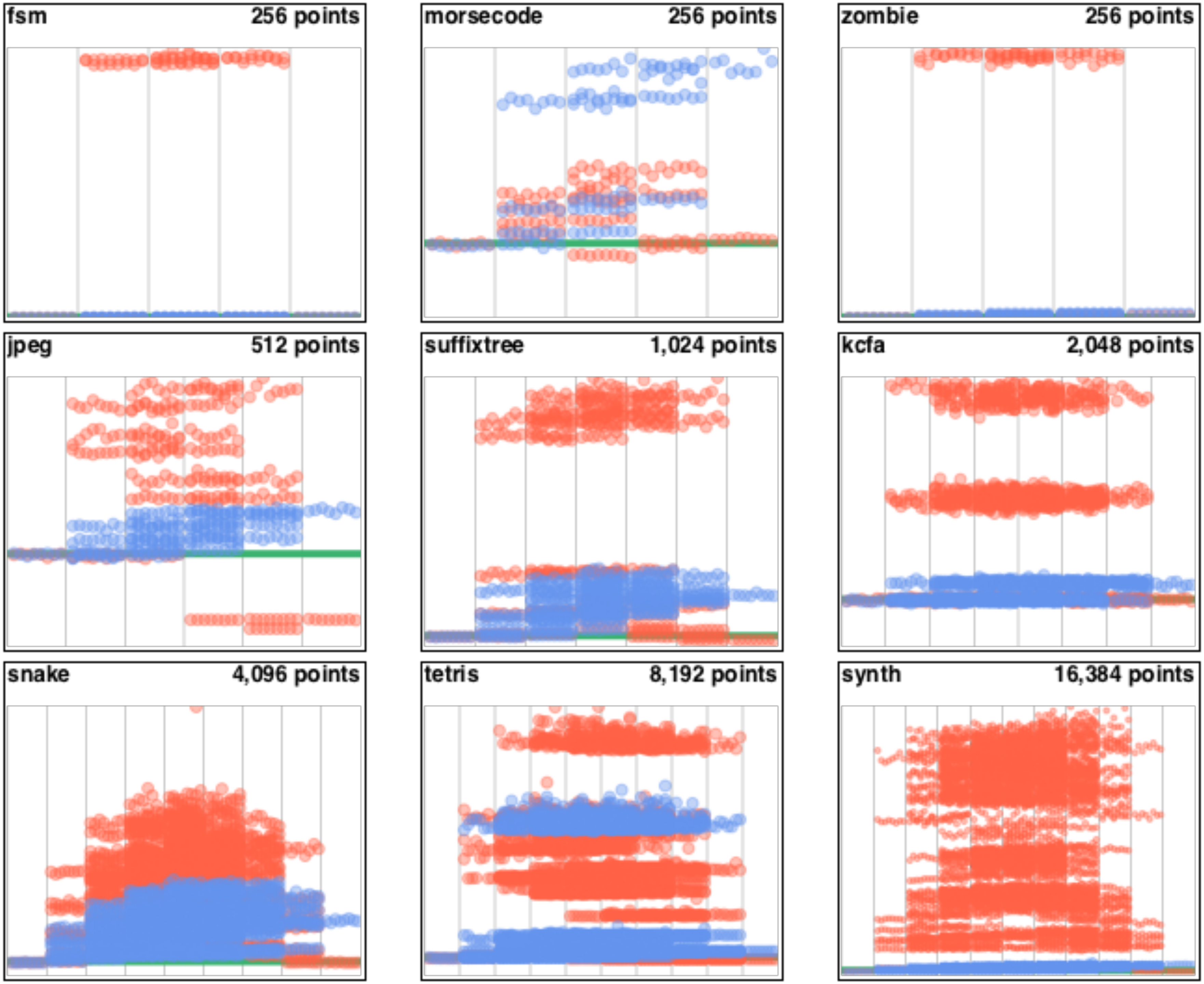


first-order (enforce type constructors)



erasure (ignore types)

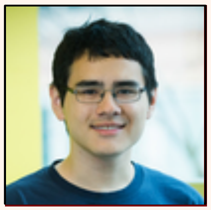




fsm
256 points
morsecode
256 points
zombie
256 points
jpeg
512 points
suffixtree
1,024 points
kcfa
2,048 points
snake
4,096 points
tetris
8,192 points
synth
16,384 points

ECOOP 2018

KafKa: Gradual Typing for Objects


Benjamin Chung


Francesco Zappa Nardelli


Paley Li


Jan Vitek