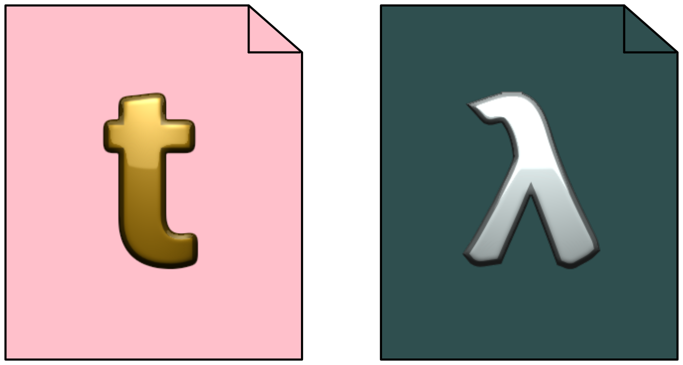
A Spectrum of Type Soundness and Performance

Ben Greenman & Matthias Felleisen
Northeastern University

Is type soundness all-or-nothing?

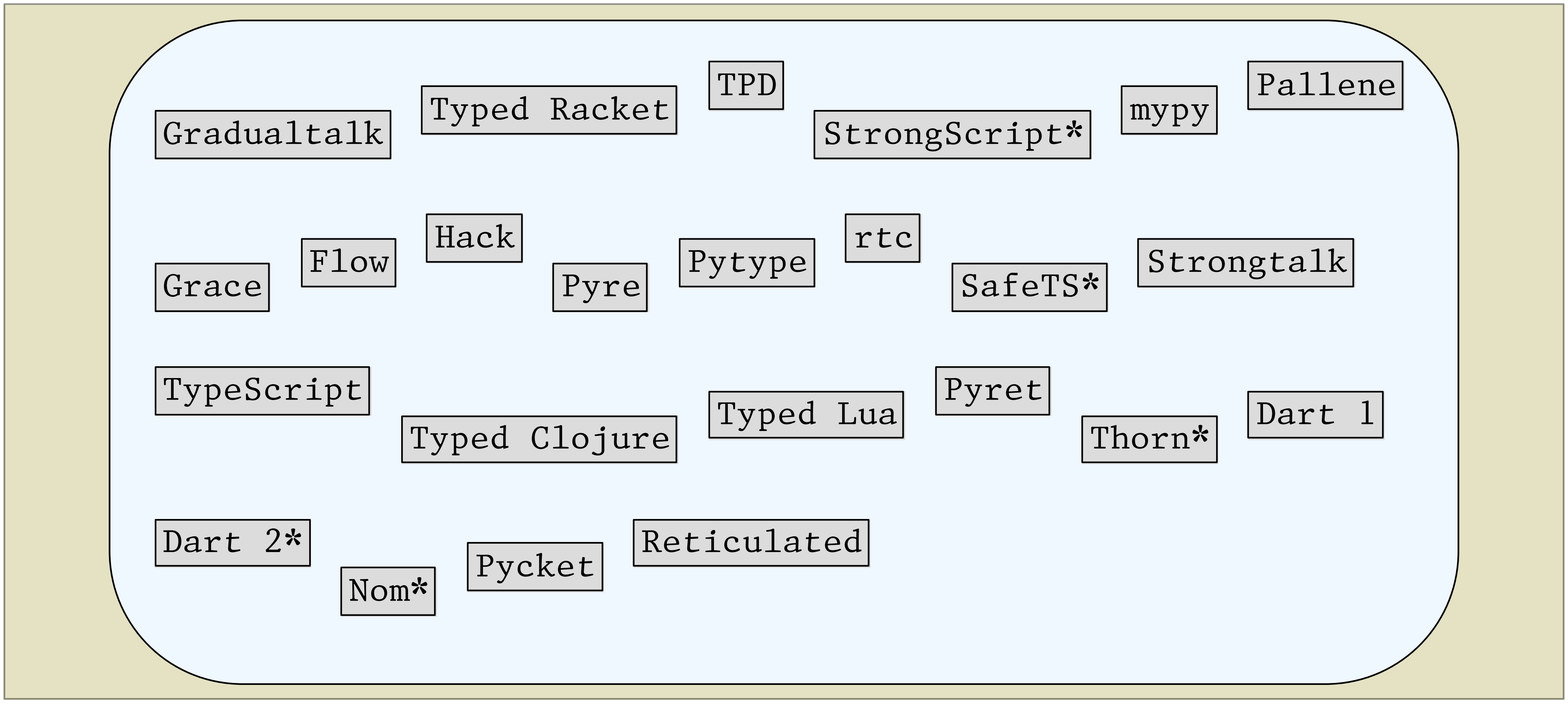
Can adding types slow down a program?

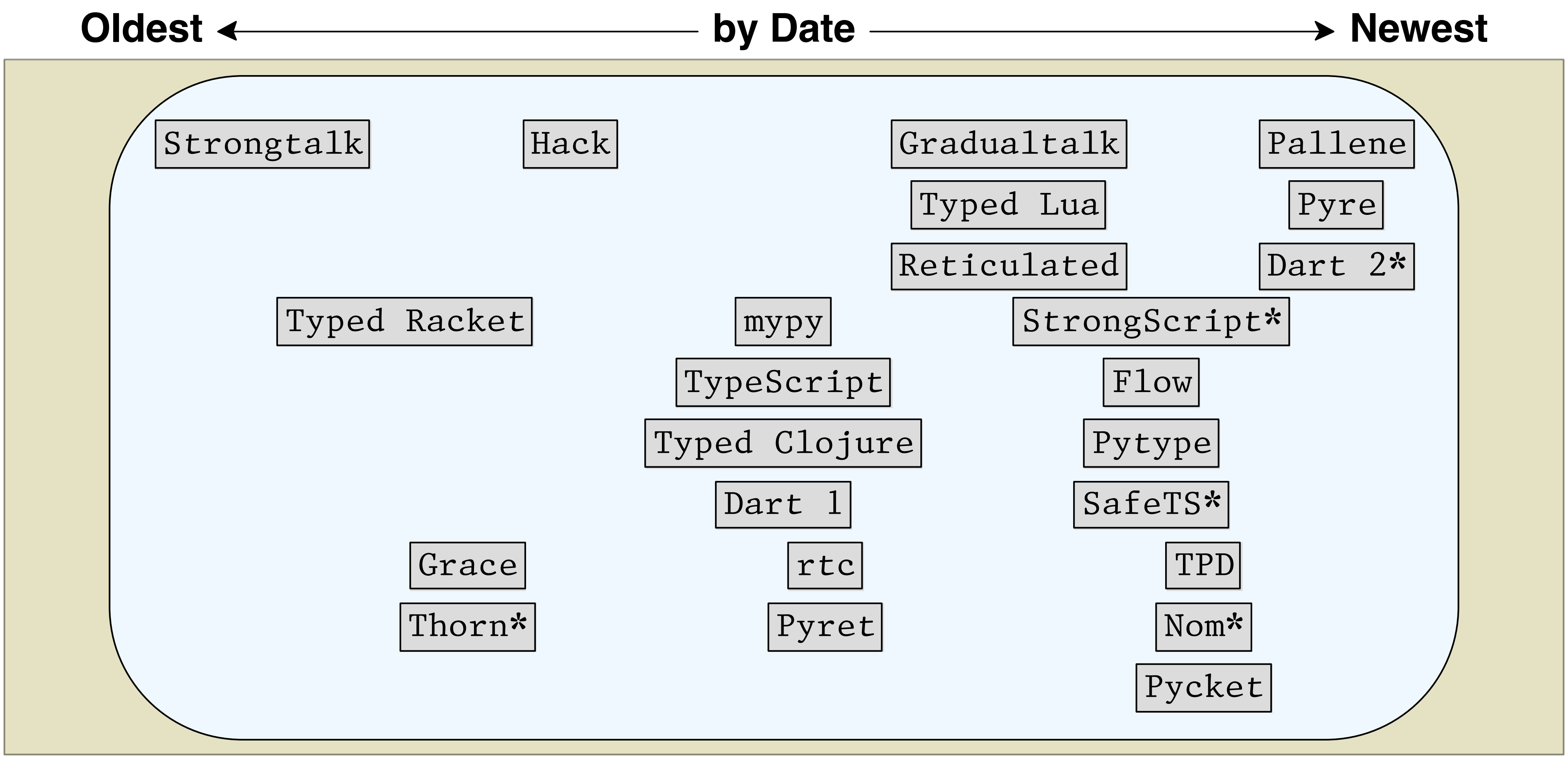
Migratory Typing

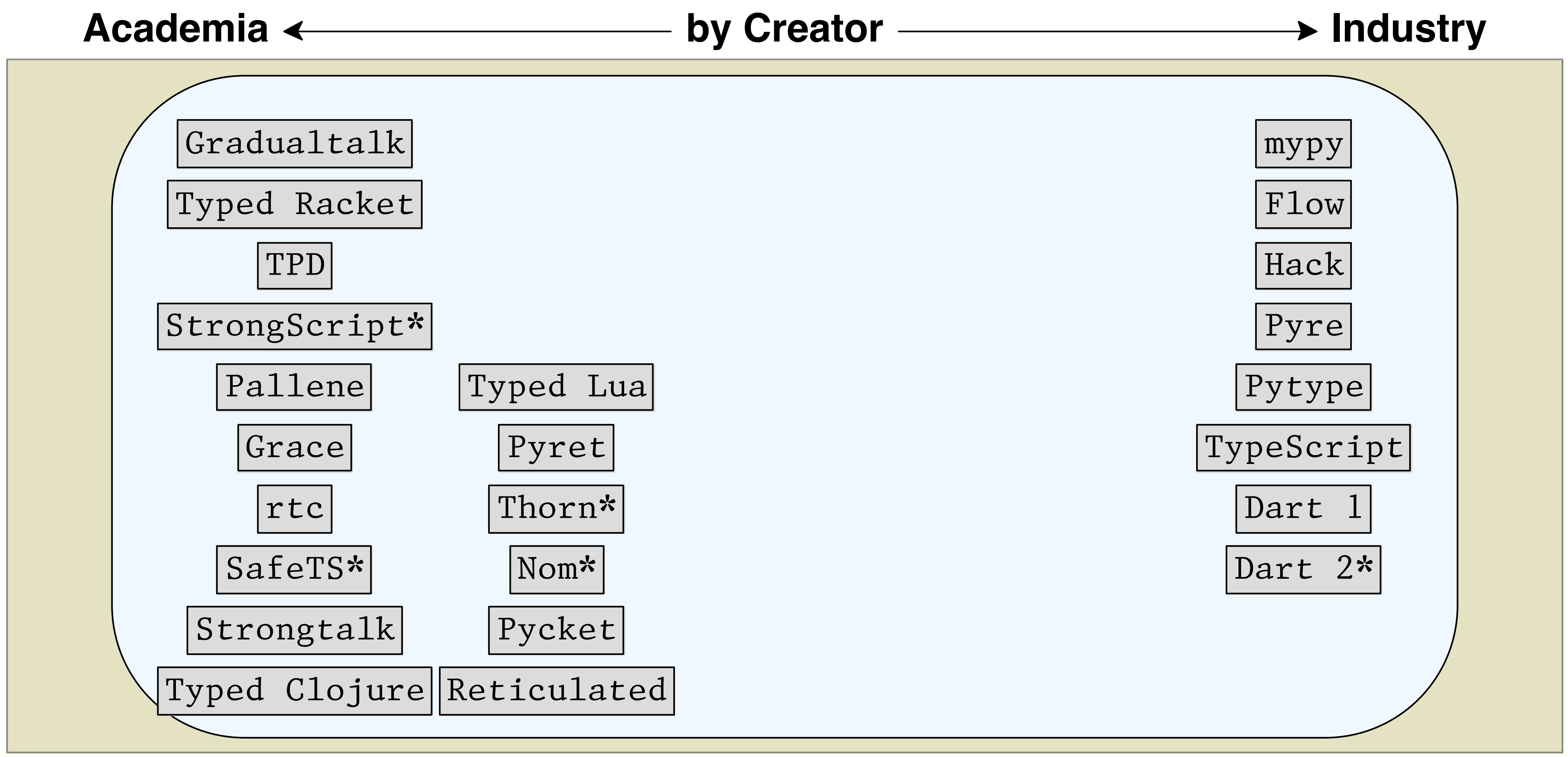

t
λ

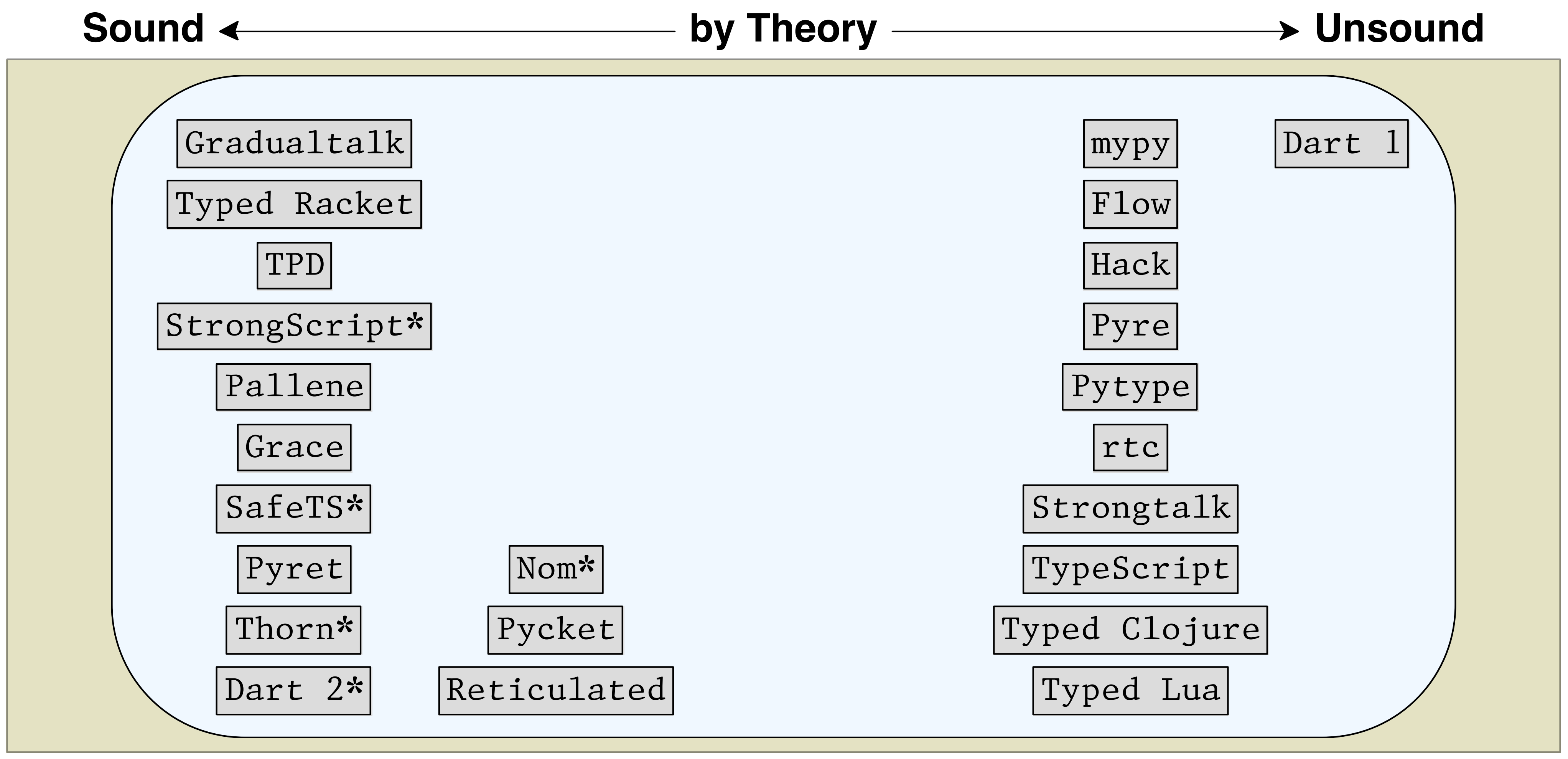
Typed/Untyped Languages

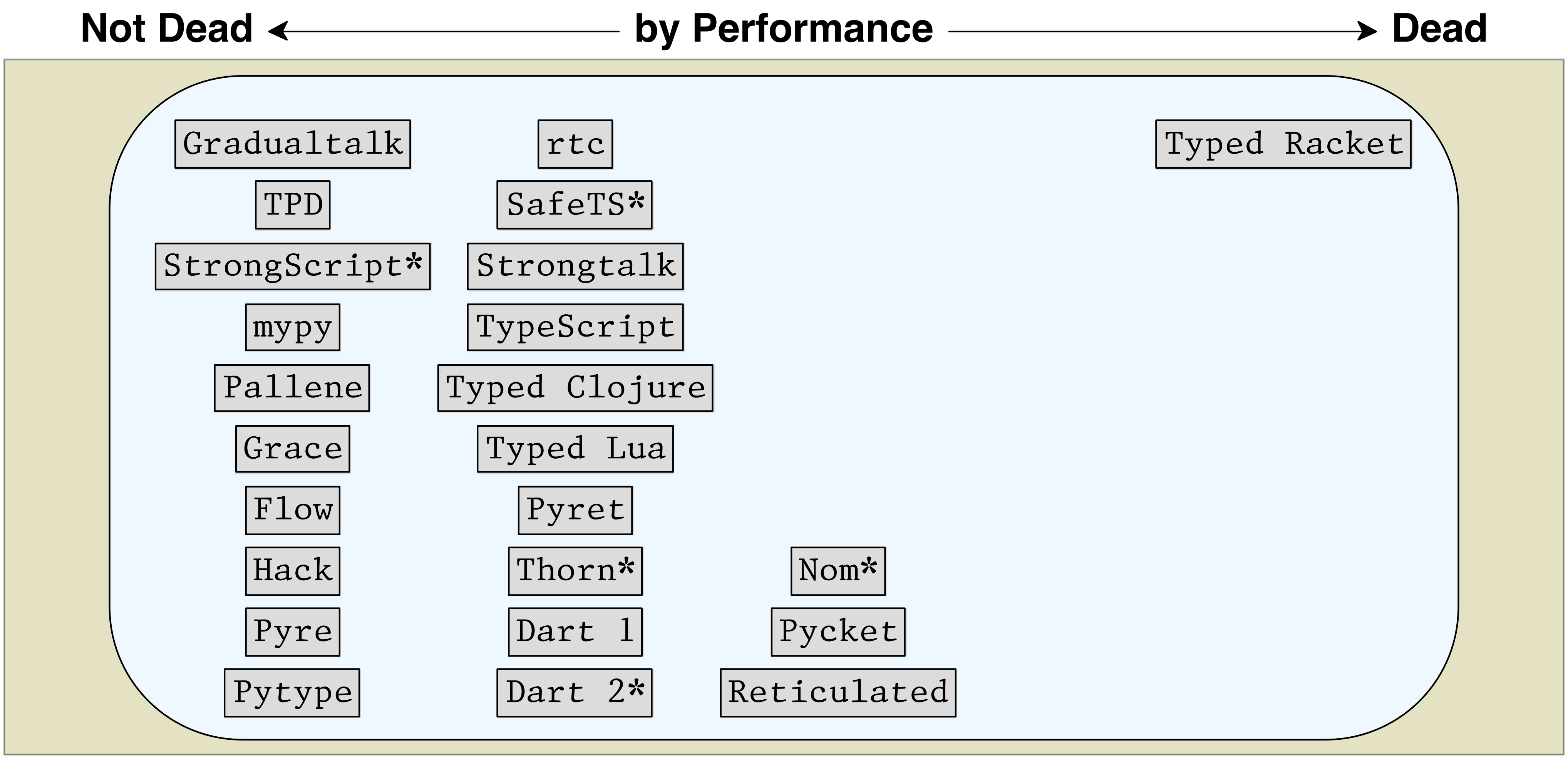
Typed/Untyped Languages




Oldest
by Date
Newest
Strongtalk
Hack
Gradualtalk
Pallene
Typed Lua
Pyre
Reticulated
Dart 2*
Typed Racket
mypy
StrongScript*
TypeScript
Flow
Typed Clojure
Pytype
Dart 1
SafeTS*
Grace
rtc
TPD
Thorn*
Pyret
Nom*
Pycket


Academia
by Creator
Industry
Gradualtalk
Typed Racket
TPD
StrongScript*
Pallene
Grace
rtc
SafeTS*
Strongtalk
Typed Clojure
Typed Lua
Pyret
Thorn*
Nom*
Pycket
Reticulated
mypy
Flow
Hack
Pyre
Pytype
TypeScript
Dart 1
Dart 2*

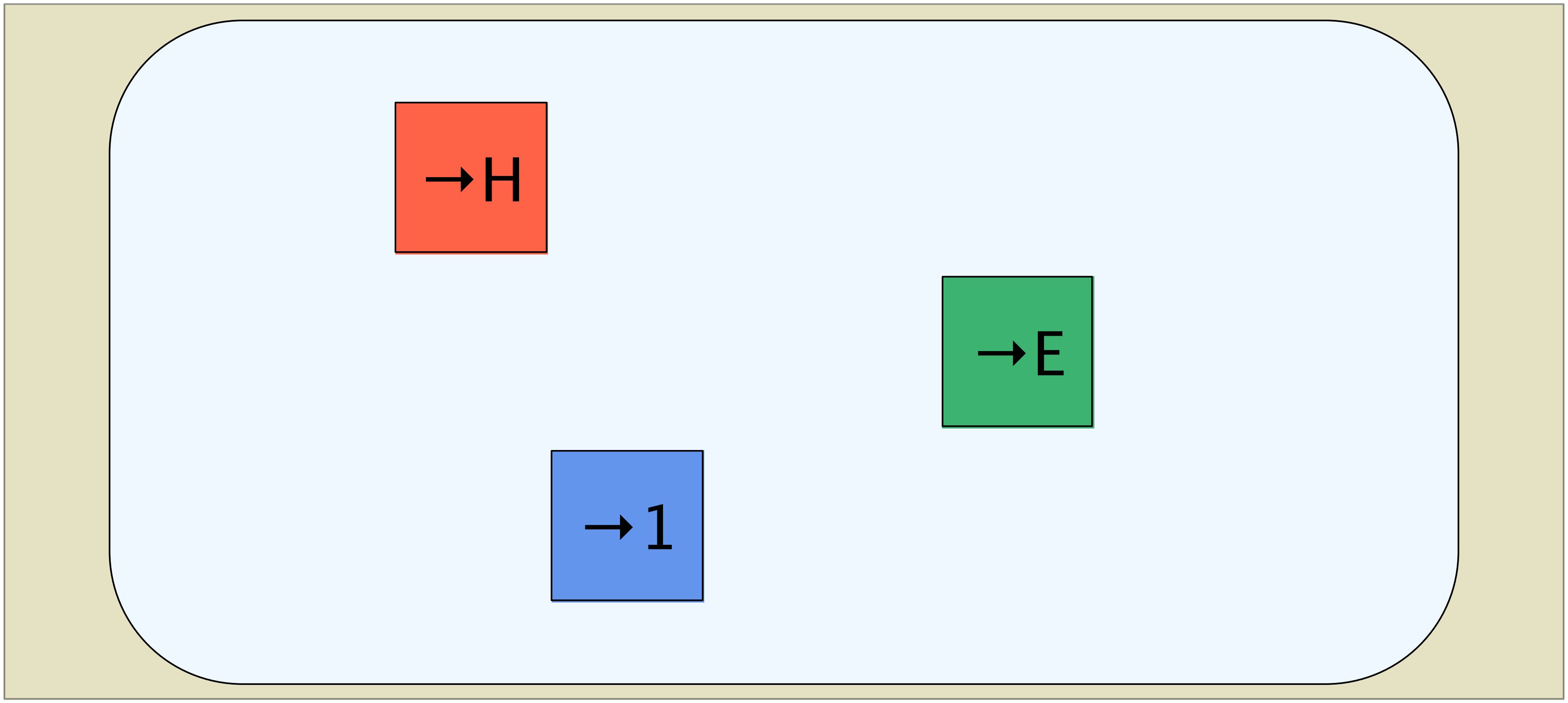

Sound
by Theory
Unsound
Gradualtalk
Typed Racket
TPD
StrongScript*
Pallene
Grace
SafeTS*
Pyret
Thorn*
Dart 2*
Nom*
Pycket
Reticulated
mypy
Flow
Hack
Pyre
Pytype
rtc
Strongtalk
TypeScript
Typed Clojure
Typed Lua
Dart 1


Not Dead
by Performance
Dead
Gradualtalk
TPD
StrongScript*
mypy
Pallene
Grace
Flow
Hack
Pyre
Pytype
rtc
SafeTS*
Strongtalk
TypeScript
Typed Clojure
Typed Lua
Pyret
Thorn*
Dart 1
Dart 2*
Nom*
Pycket
Reticulated
Typed Racket

Chaos!

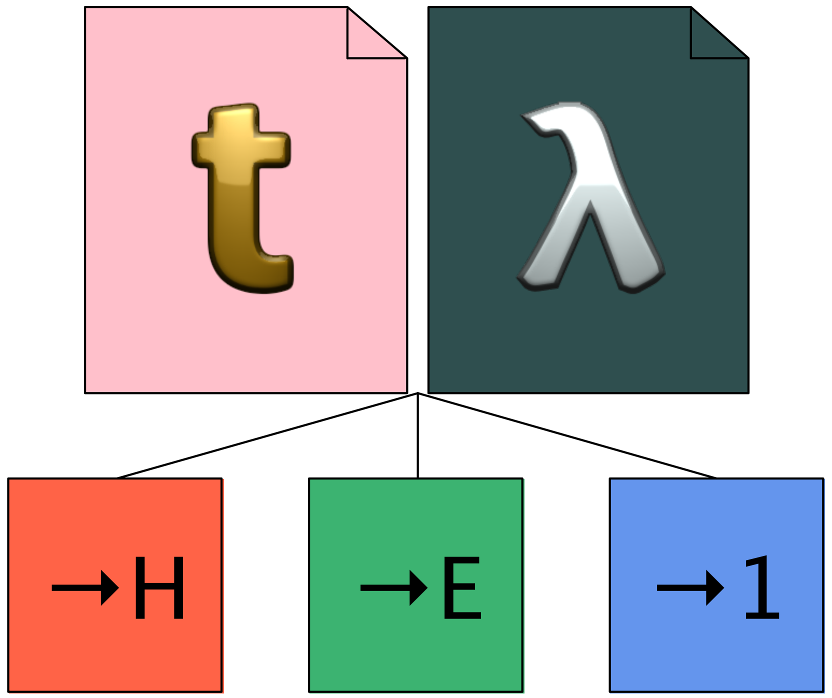
KafKa: Gradual Typing for Objects

Who	*Benjamin W Chung*, *Paley Li*, *Francesco Zappa Nardelli*, *Jan Vitek*
Track	ECOOP 2018 ECOOP Research Papers


→H
→E
→1

🟥 $\rightarrow_H$	higher-order semantics
🟩 $\rightarrow_E$	erasure semantics
🟦 $\rightarrow_1$	first-order semantics

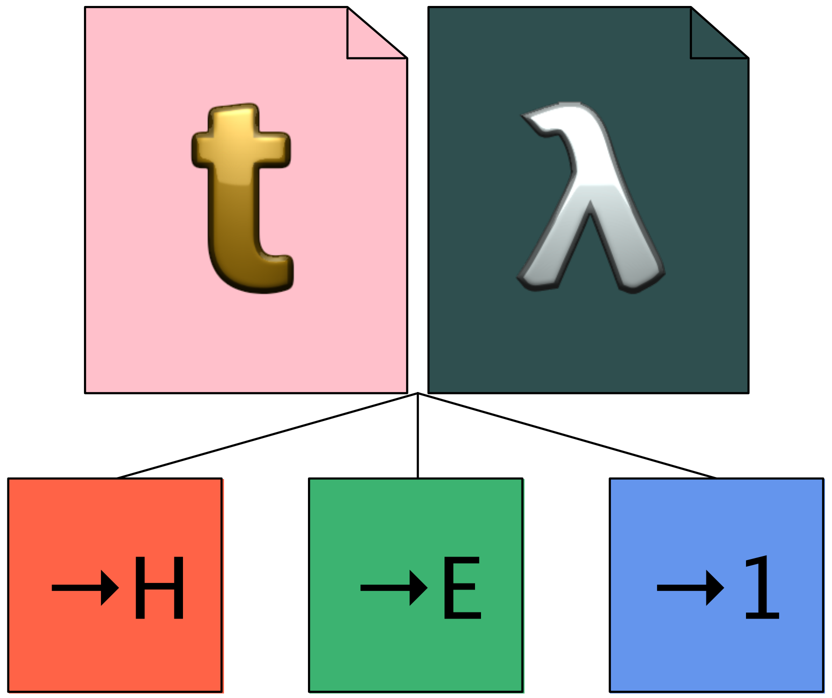
Contributions (1/2)



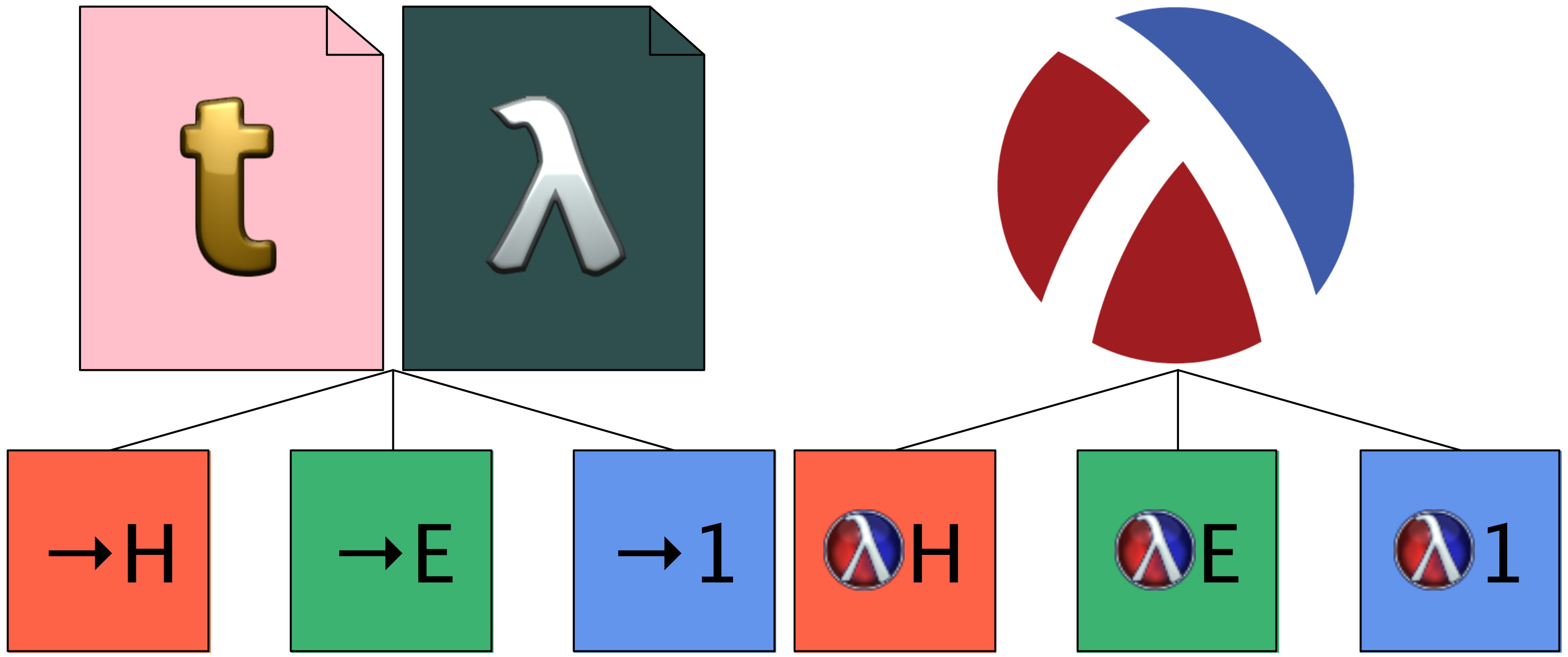
Model:

- one mixed-typed language
- one surface type system
- three semantics

Contributions (2/2)



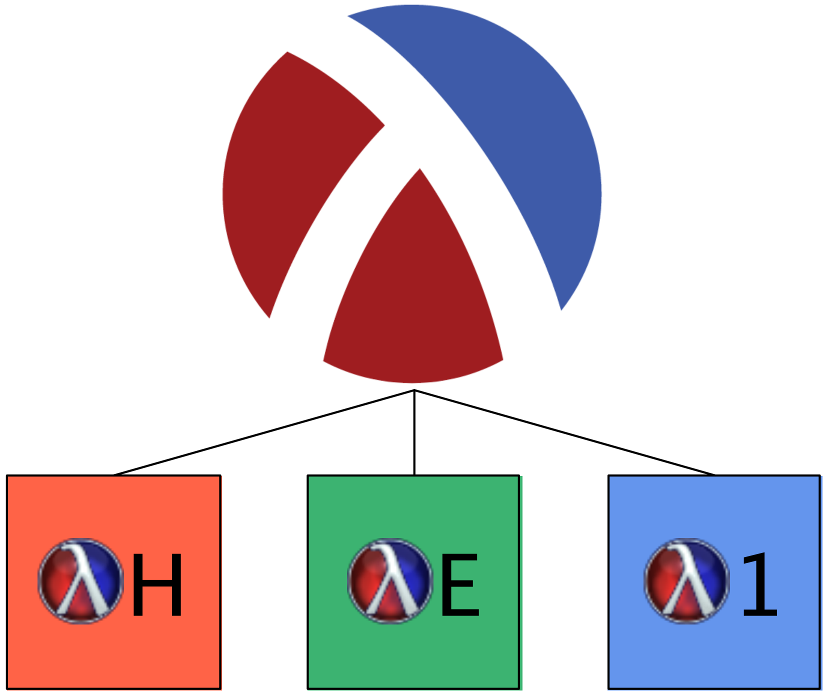
Contributions (2/2)



Contributions (2/2)

Implementation:

- Racket syntax/types
- three compilers
- the first controlled performance experiment



Model

t = Nat | Int | $t \times t$ | $t \to t$

Nat <: Int

t = Nat | Int | $t \times t$ | $t \to t$

Nat <: Int

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

n $\subset$ i

t = Nat | Int | $t \times t$ | $t \to t$

Nat <: Int

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

n $\subset$ i

e = ... | dyn t e | stat t e

```
t = Nat | Int | t×t | t → t
Nat <: Int

v = n | i | ⟨v,v⟩ | λ(x)e | λ(x:t)e

n ⊂ i

e = .... | dyn t e | stat t e
```

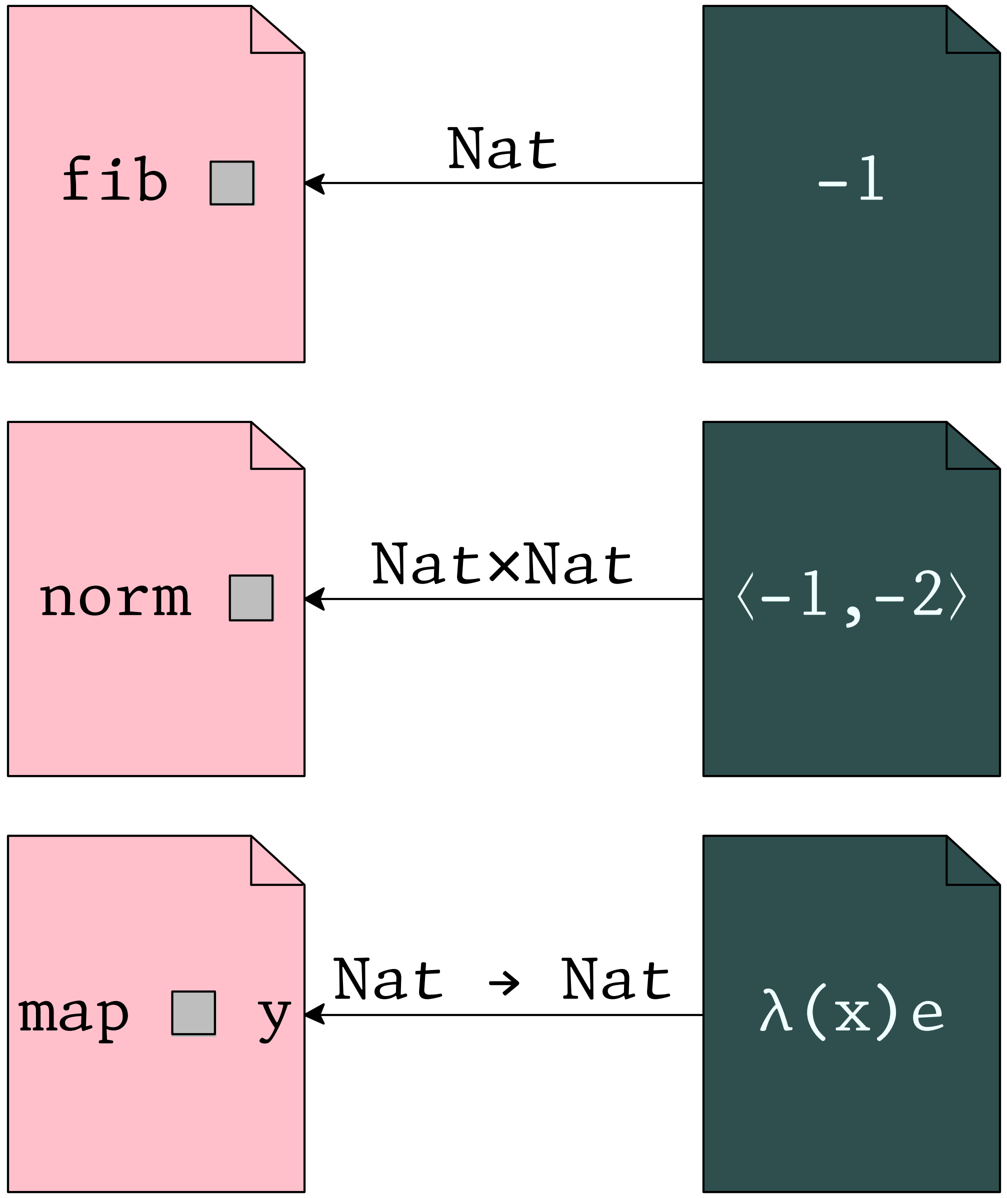
$$\frac{\vdash e}{\vdash \texttt{dyn}\ t\ e : t} \qquad \frac{\vdash e : t}{\vdash \texttt{stat}\ t\ e}$$

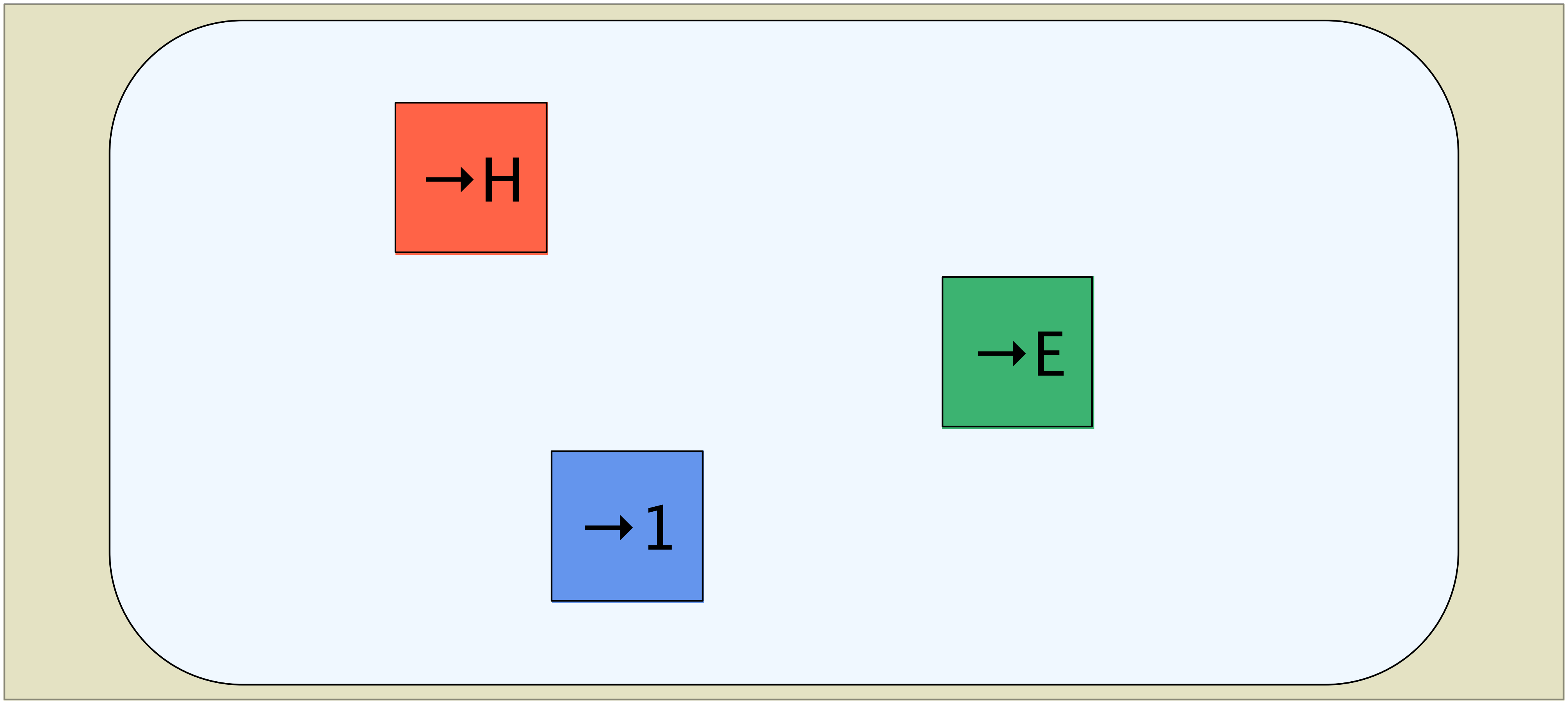
$$\Gamma = \begin{cases} \texttt{fib : Nat → Nat} \\ \texttt{norm : Nat×Nat → Nat} \\ \texttt{map : (Nat → Nat) → Nat×Nat → Nat×Nat} \end{cases}$$

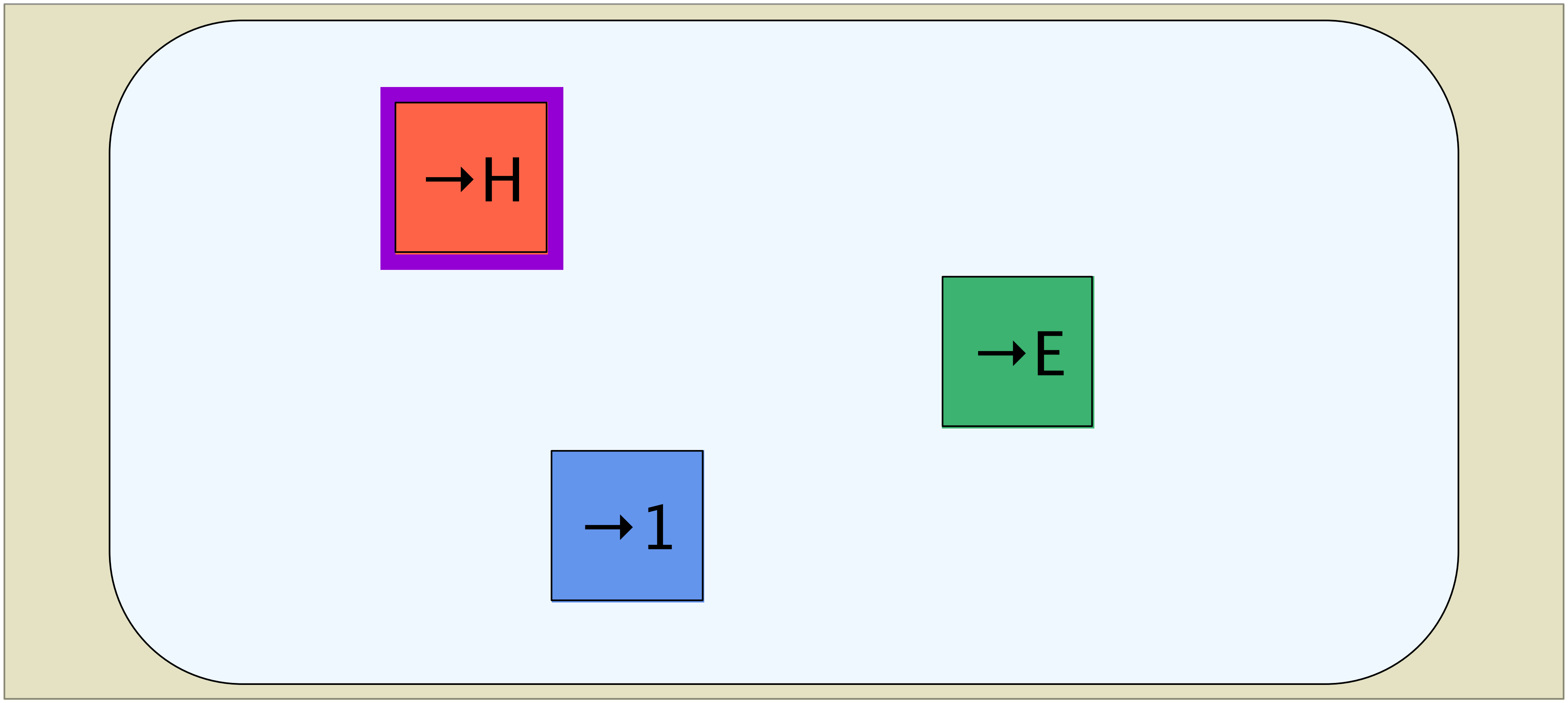
$\Gamma \vdash$ `fib (dyn Nat -1) : Nat`

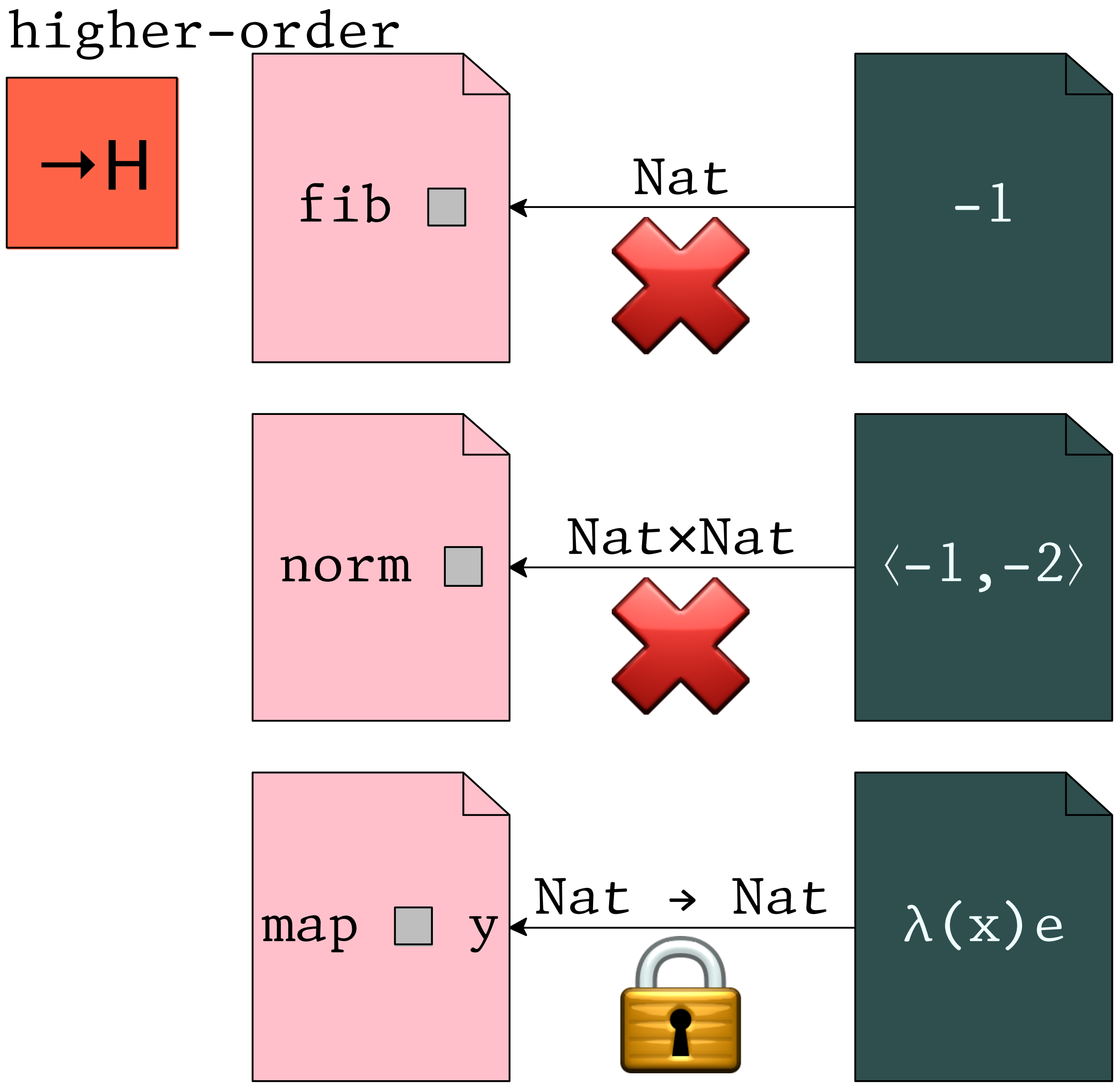
$\Gamma \vdash$ `norm (dyn Nat×Nat ⟨-1,-2⟩) : Nat`

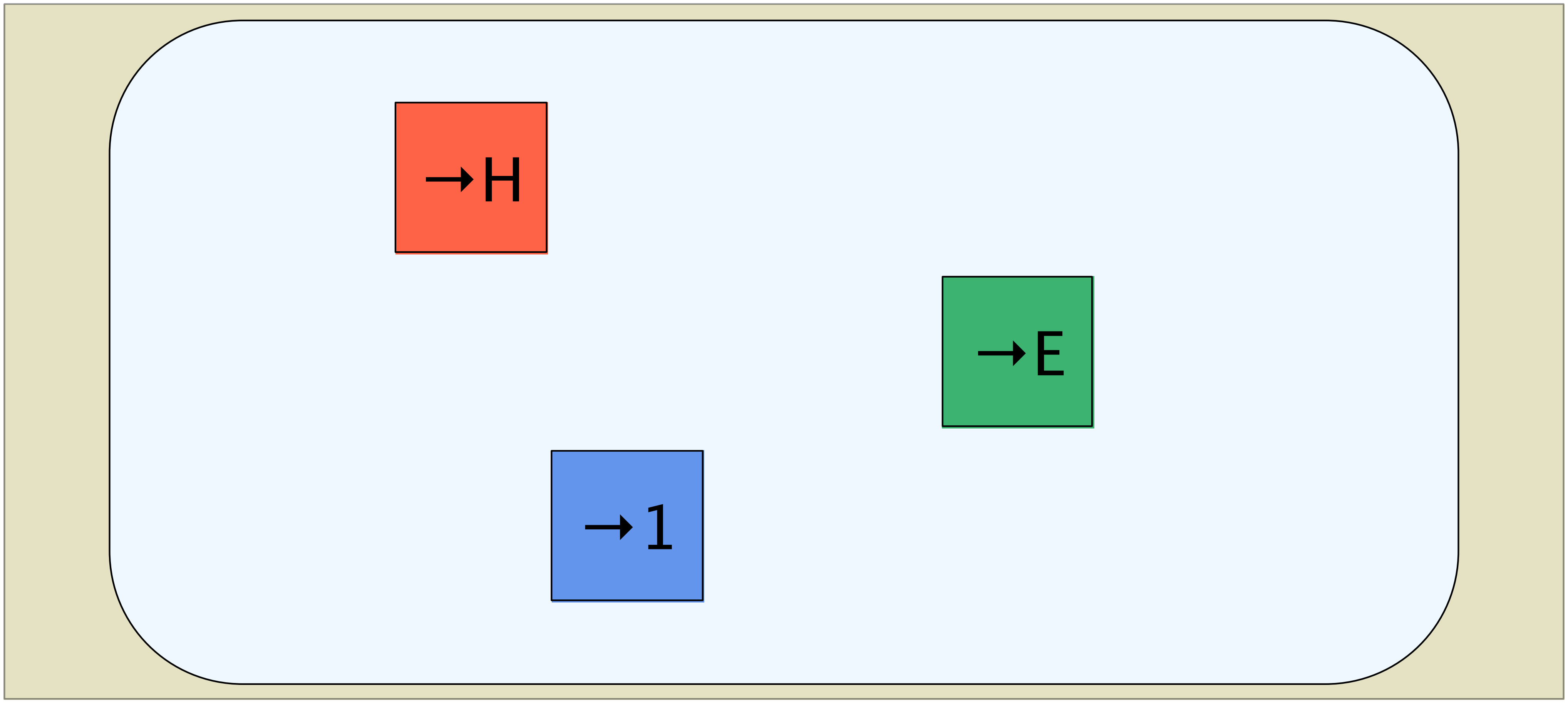
$\Gamma \vdash$ `map (dyn (Nat → Nat) (λ(x)e)) y : Nat×Nat`

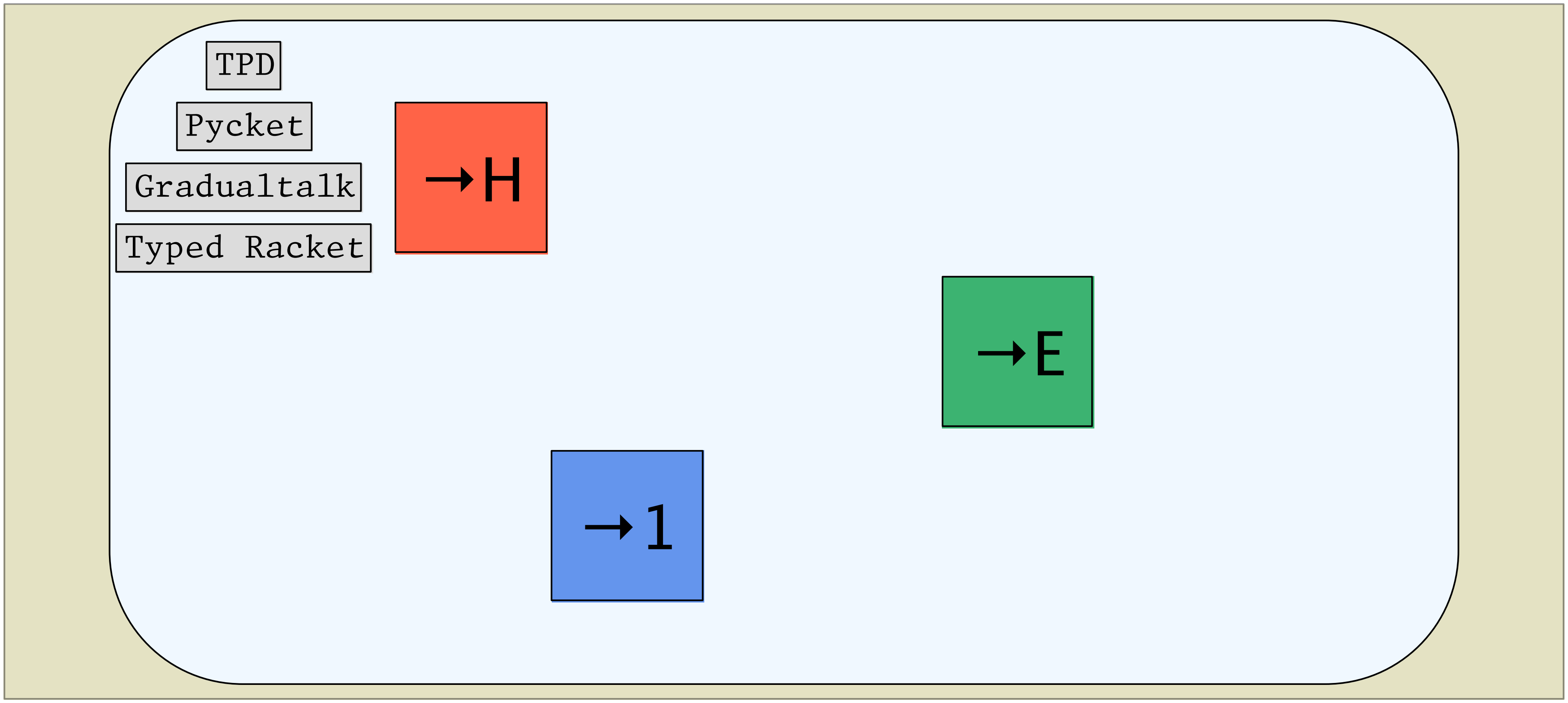

fib ◻
Nat
-1
norm ◻
Nat×Nat
⟨-1,-2⟩
map ◻ y
Nat → Nat
λ(x)e

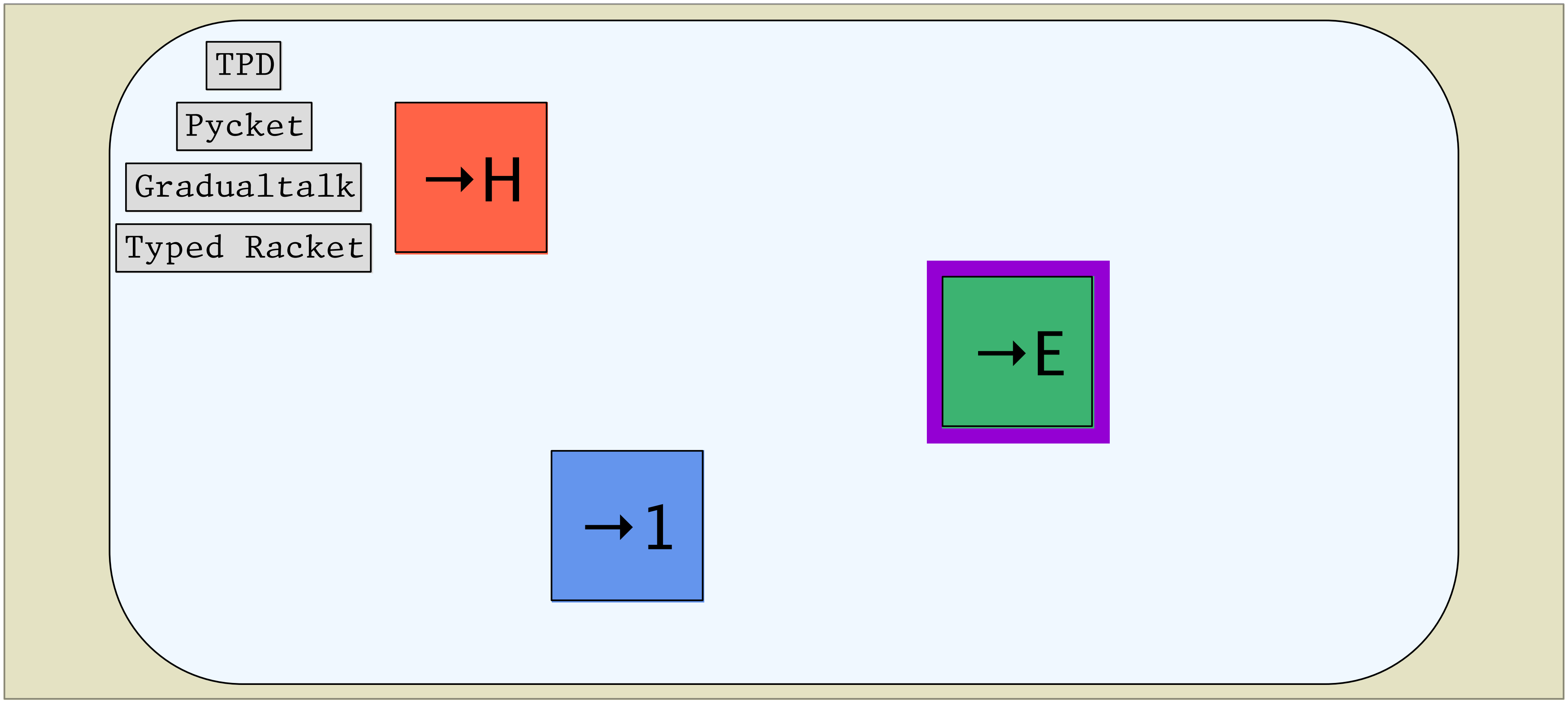

→H
→E
→1

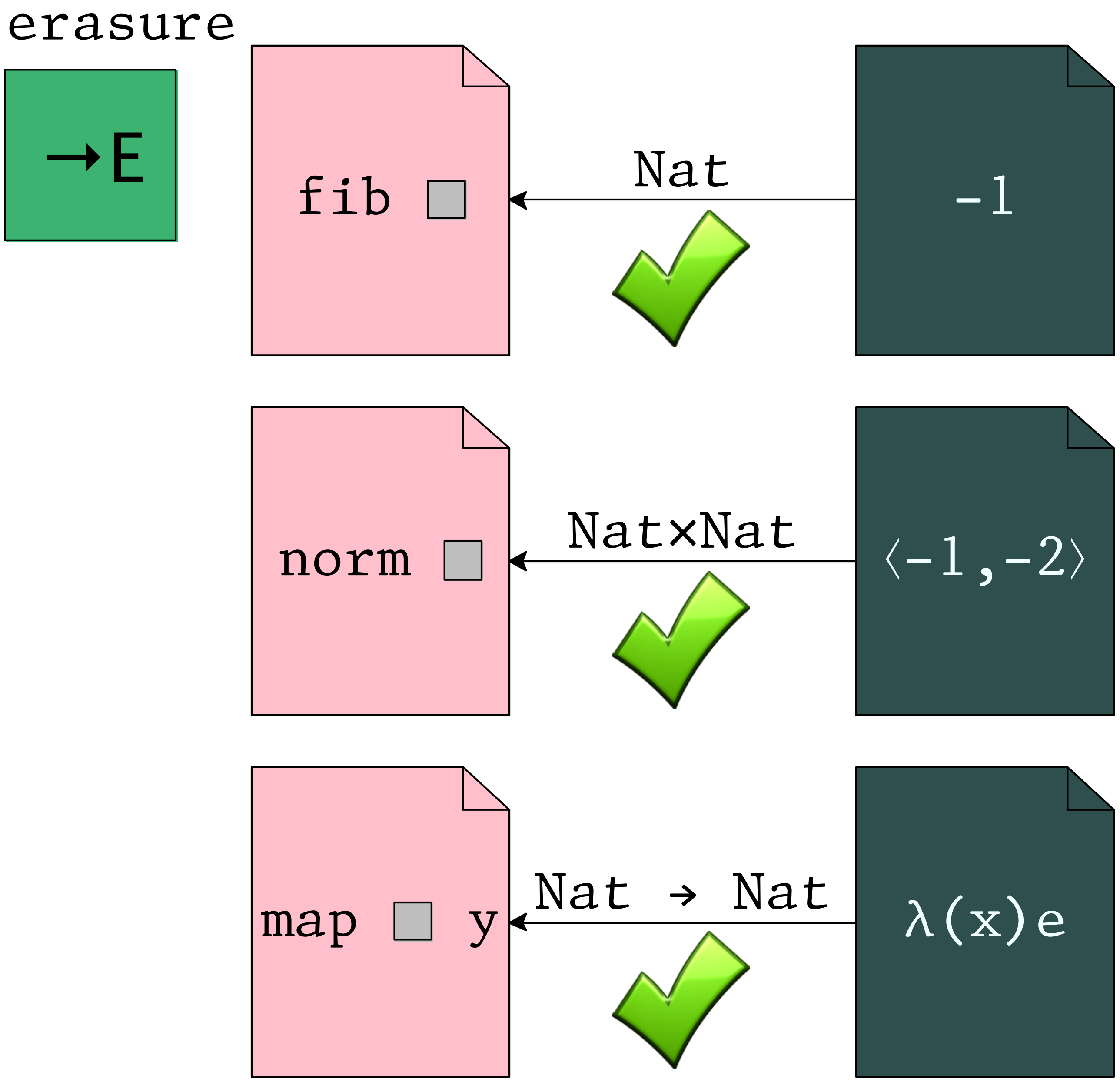

→H
→E
→1

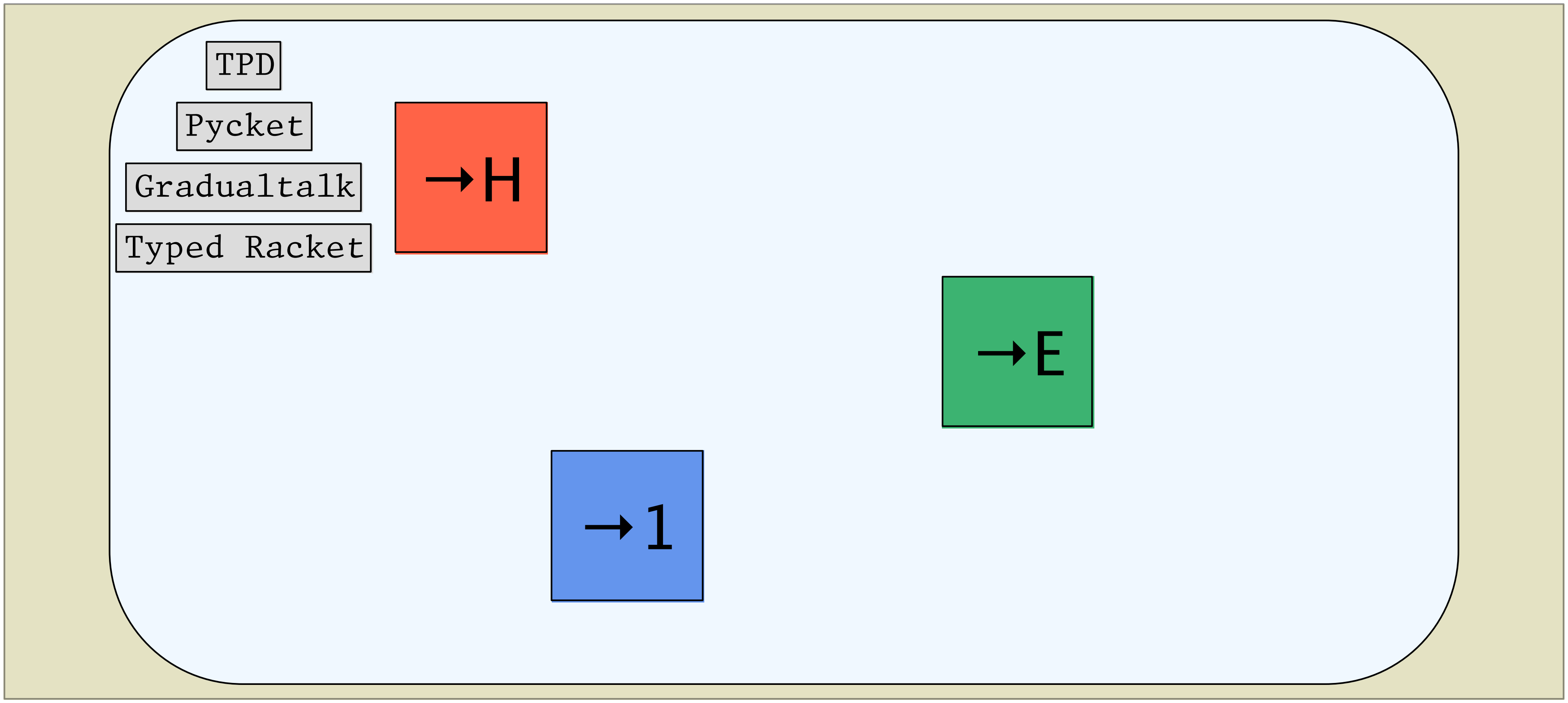

higher-order
→H
fib □
Nat
-1
norm □
Nat×Nat
⟨-1,-2⟩
map □ y
Nat → Nat
λ(x)e

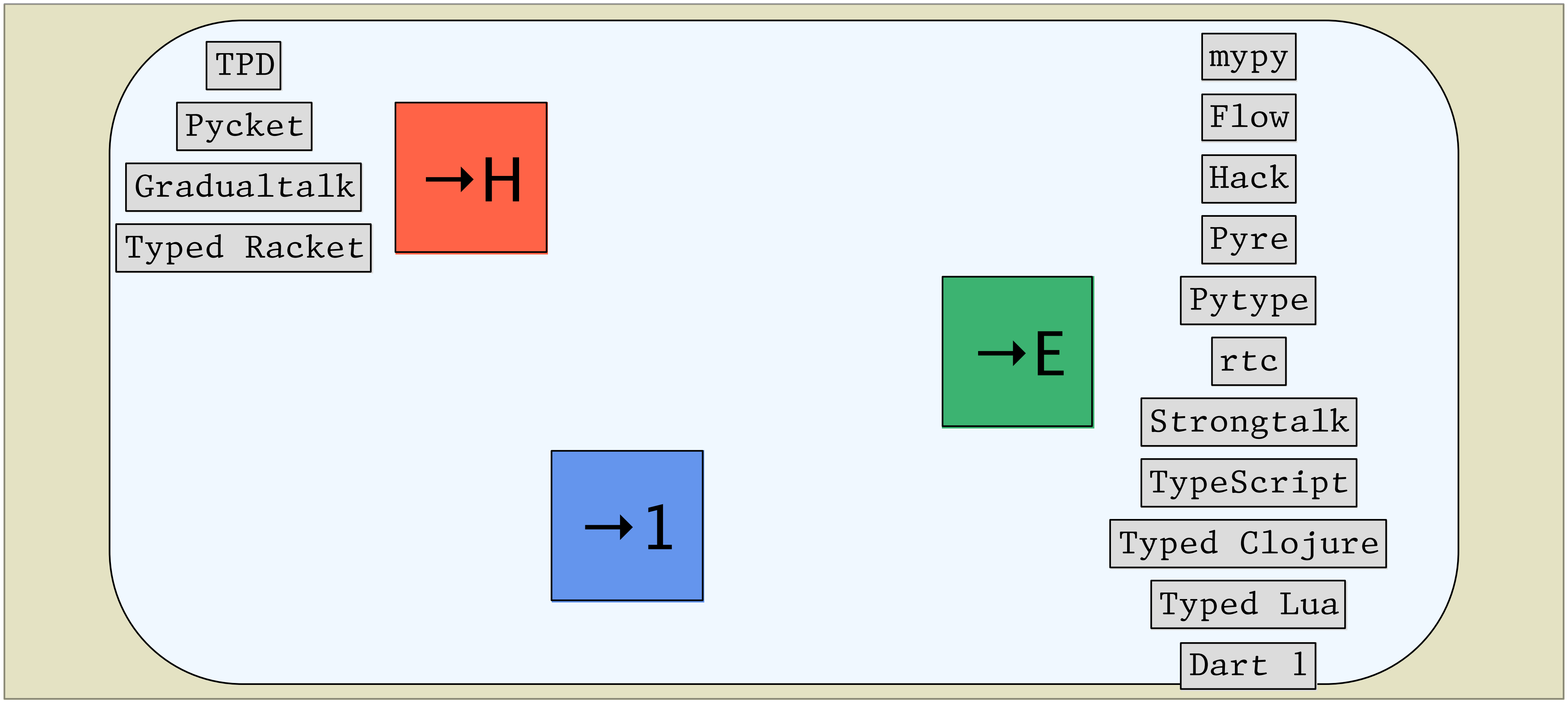

→H
→E
→1

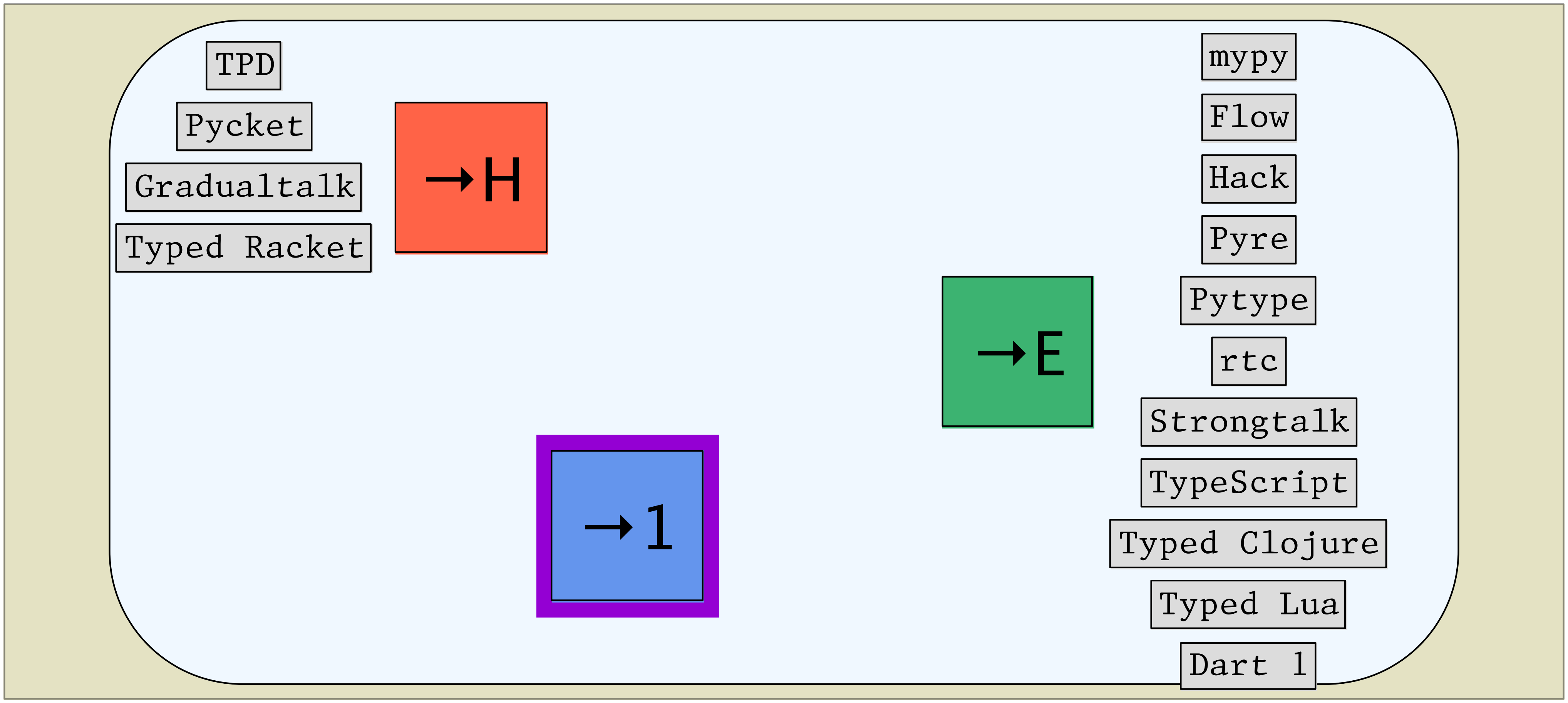

TPD
Pycket
Gradualtalk
Typed Racket
→H
→E
→1

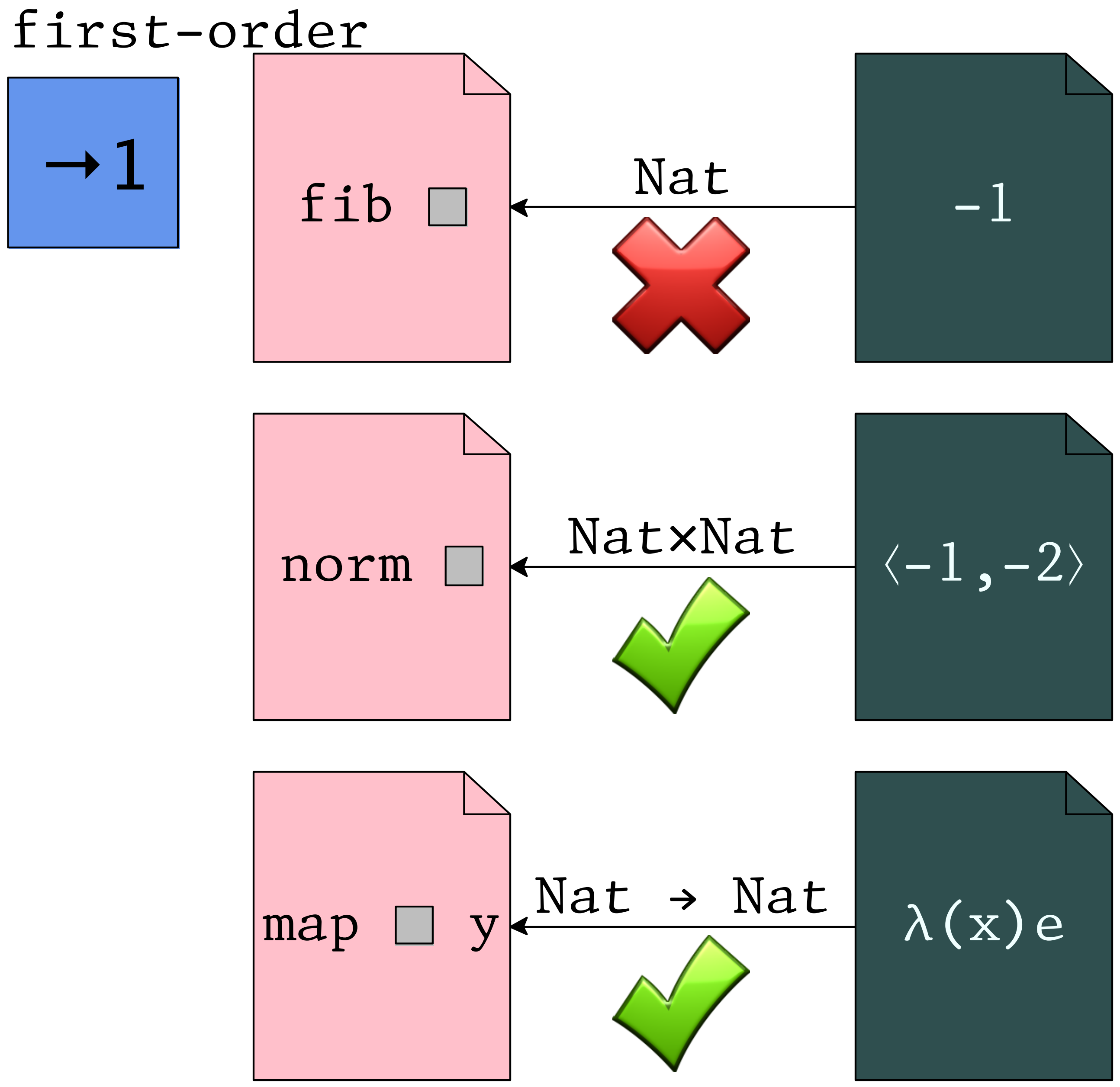

TPD
Pycket
Gradualtalk
Typed Racket
→H
→E
→1

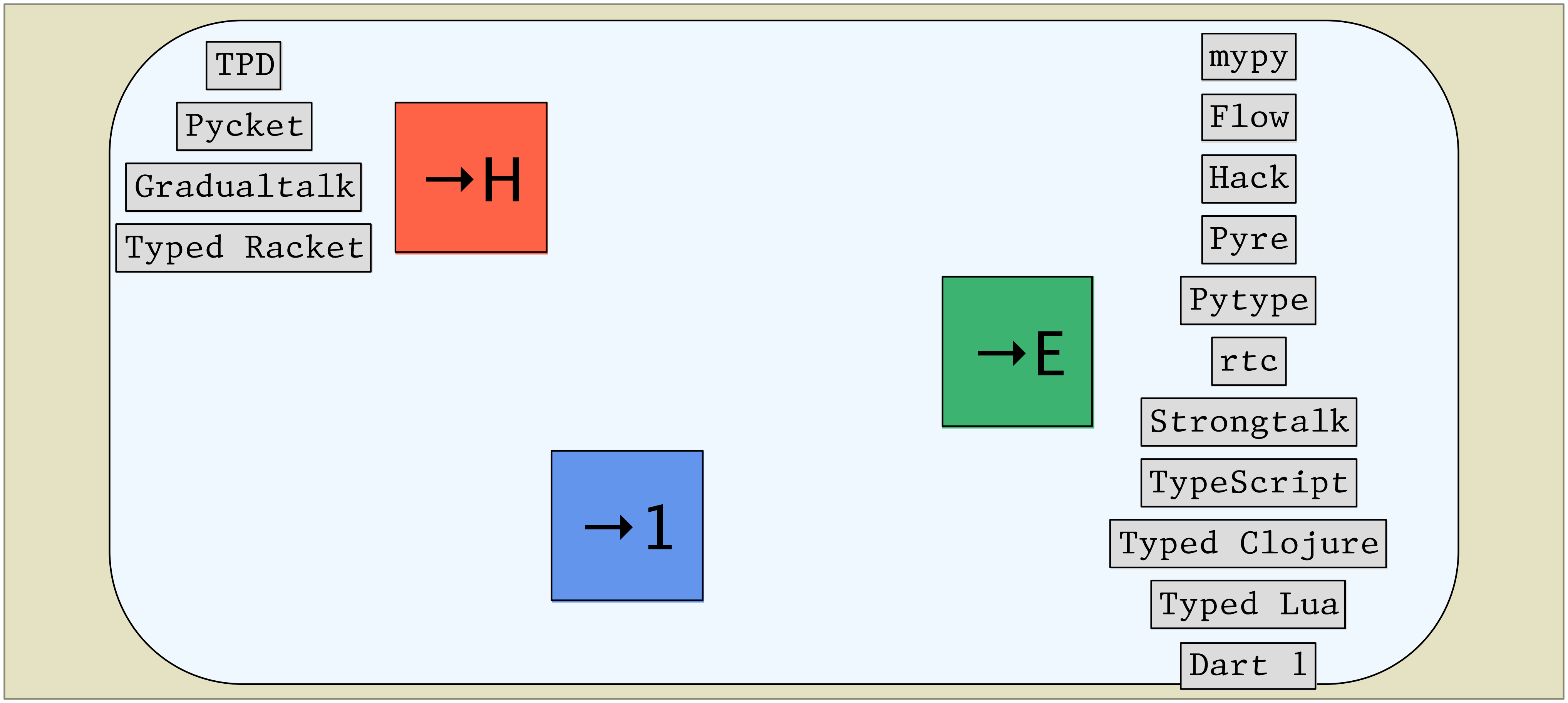

erasure
→E
fib □
Nat
-1
norm □
Nat×Nat
⟨-1,-2⟩
map □ y
Nat → Nat
λ(x)e

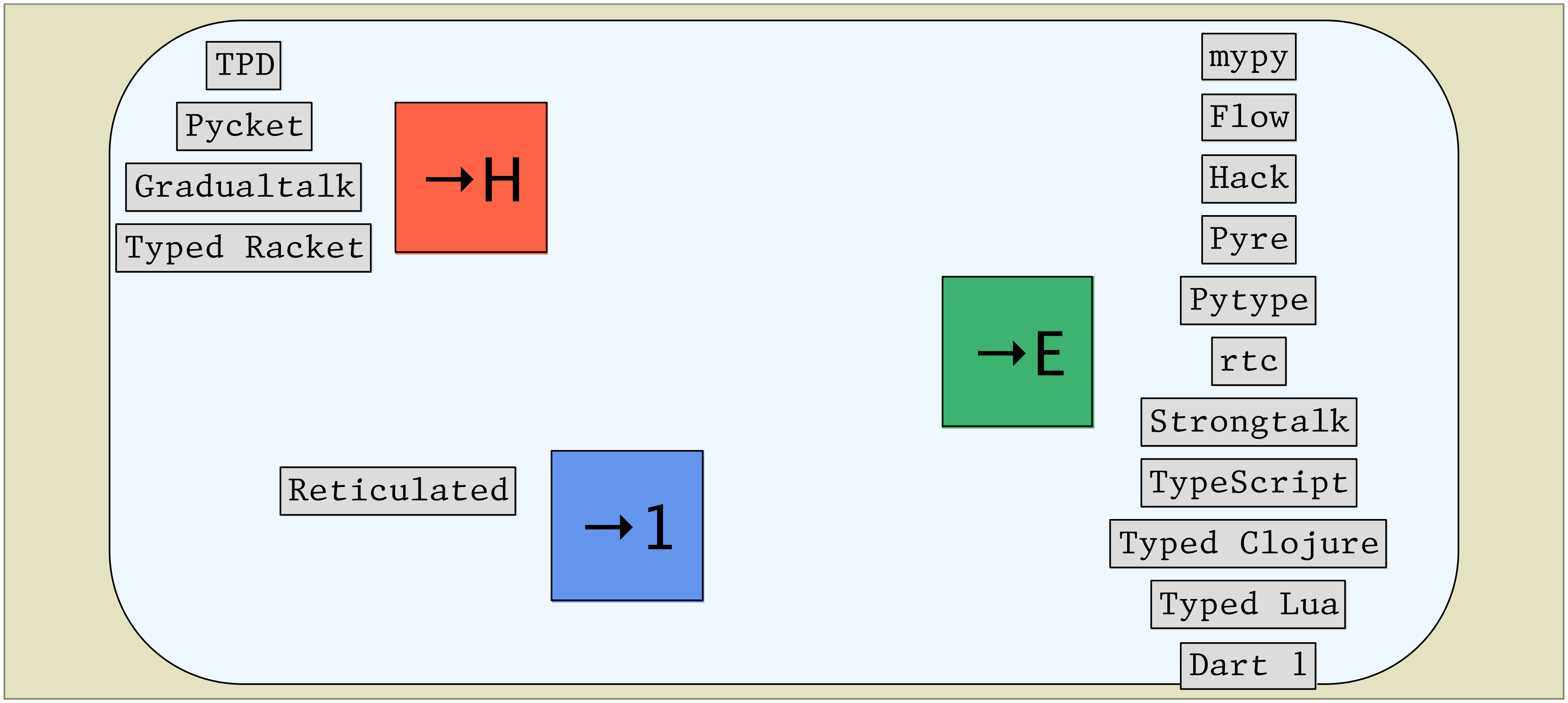

TPD
Pycket
Gradualtalk
Typed Racket
→H
→E
→1

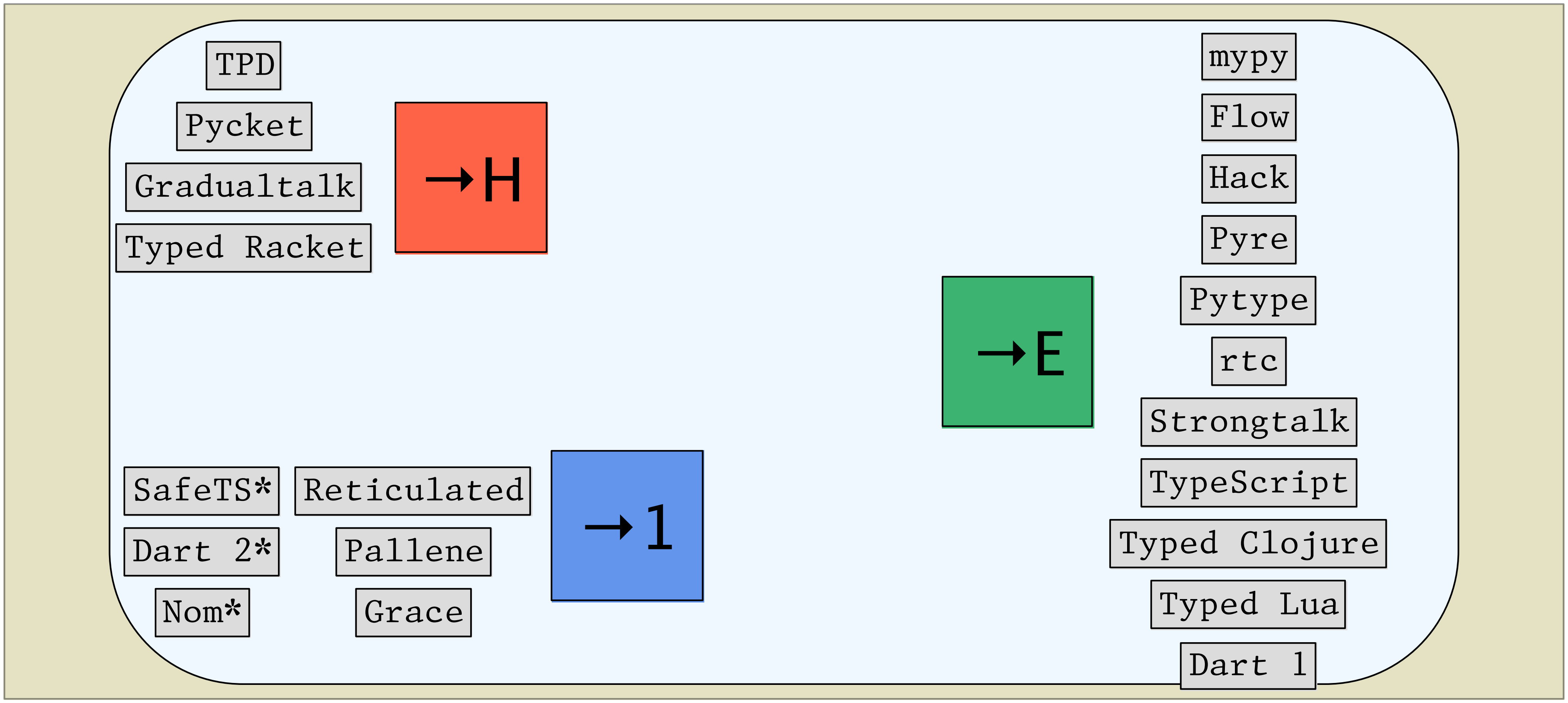

TPD
Pycket
Gradualtalk
Typed Racket
→H
→1
→E
mypy
Flow
Hack
Pyre
Pytype
rtc
Strongtalk
TypeScript
Typed Clojure
Typed Lua
Dart 1

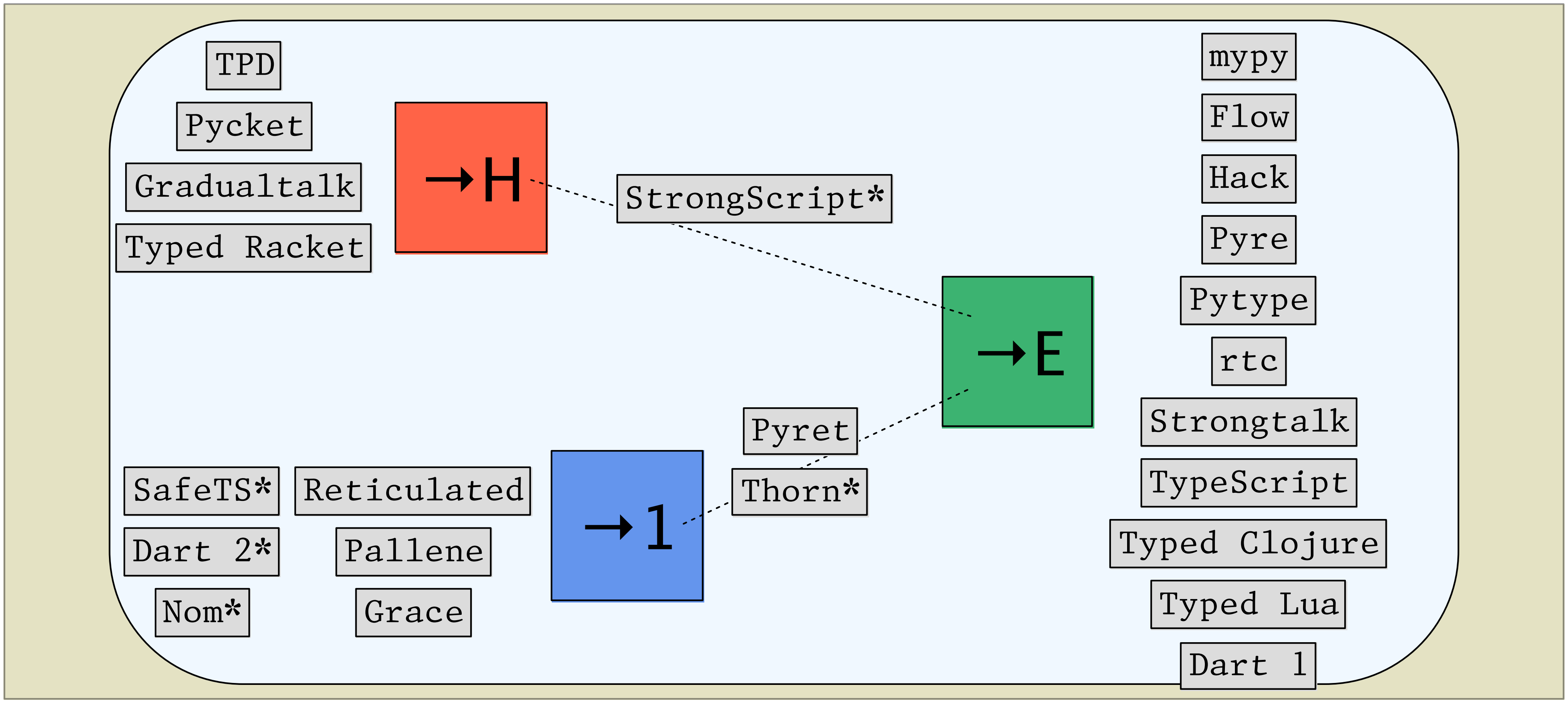

TPD
Pycket
Gradualtalk
Typed Racket
→H
mypy
Flow
Hack
Pyre
Pytype
→E
rtc
Strongtalk
TypeScript
Typed Clojure
Typed Lua
Dart 1
→1


first-order
→1
fib □
Nat
-1
norm □
Nat×Nat
⟨-1,-2⟩
map □ y
Nat → Nat
λ(x)e

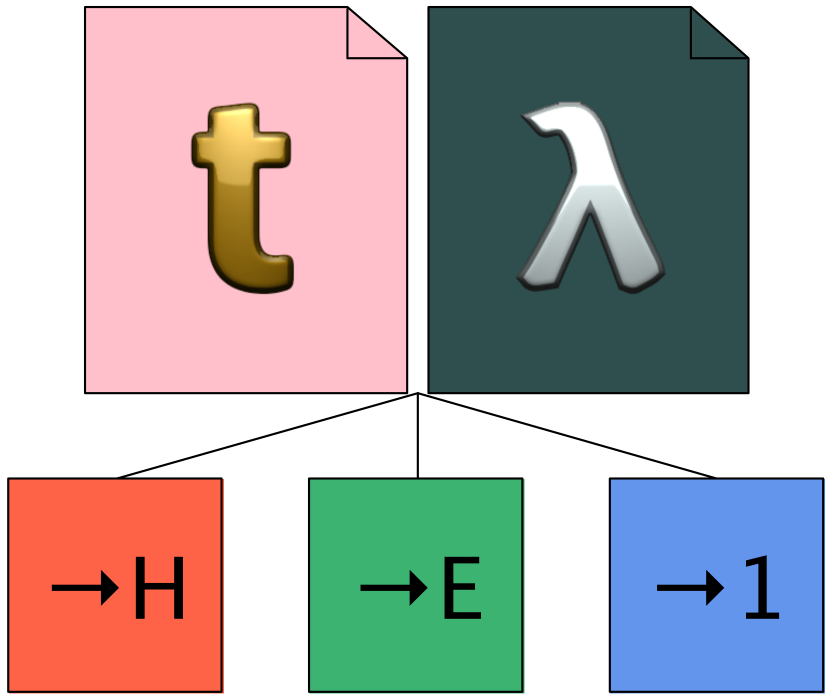

TPD
Pycket
Gradualtalk
Typed Racket
$\to H$
$\to E$
$\to 1$
mypy
Flow
Hack
Pyre
Pytype
rtc
Strongtalk
TypeScript
Typed Clojure
Typed Lua
Dart 1


TPD
Pycket
Gradualtalk
Typed Racket
→H
→E
mypy
Flow
Hack
Pyre
Pytype
rtc
Strongtalk
TypeScript
Typed Clojure
Typed Lua
Dart 1
Reticulated
→1

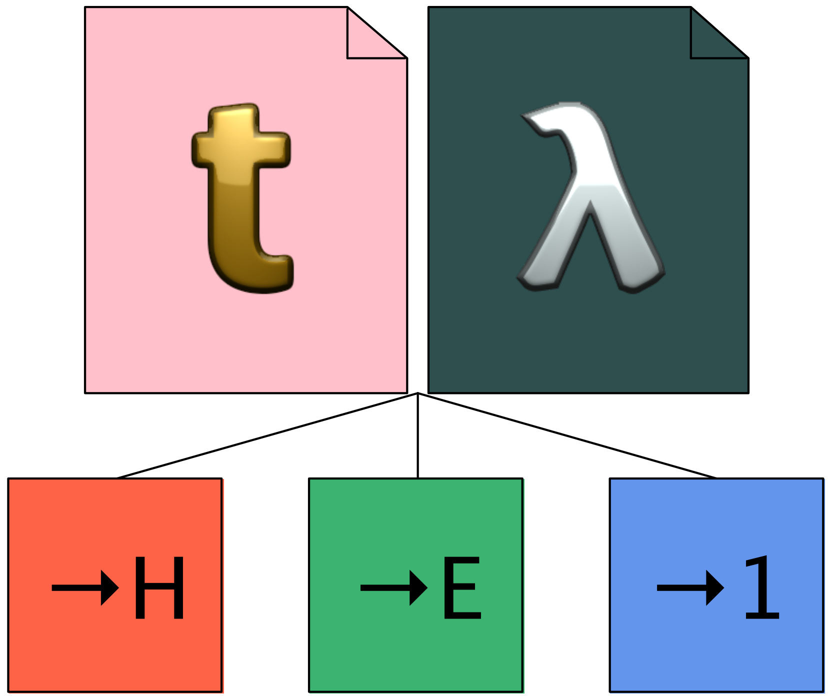

TPD
Pycket
Gradualtalk
Typed Racket
→H
mypy
Flow
Hack
Pyre
Pytype
→E
rtc
Strongtalk
TypeScript
Typed Clojure
Typed Lua
Dart 1
SafeTS*
Reticulated
Dart 2*
Pallene
Nom*
Grace
→1


TPD
Pycket
Gradualtalk
Typed Racket
→H
StrongScript*
mypy
Flow
Hack
Pyre
Pytype
rtc
Strongtalk
TypeScript
Typed Clojure
Typed Lua
Dart 1
→E
Pyret
Thorn*
→1
SafeTS*
Dart 2*
Nom*
Reticulated
Pallene
Grace

Is type soundness all-or-nothing?

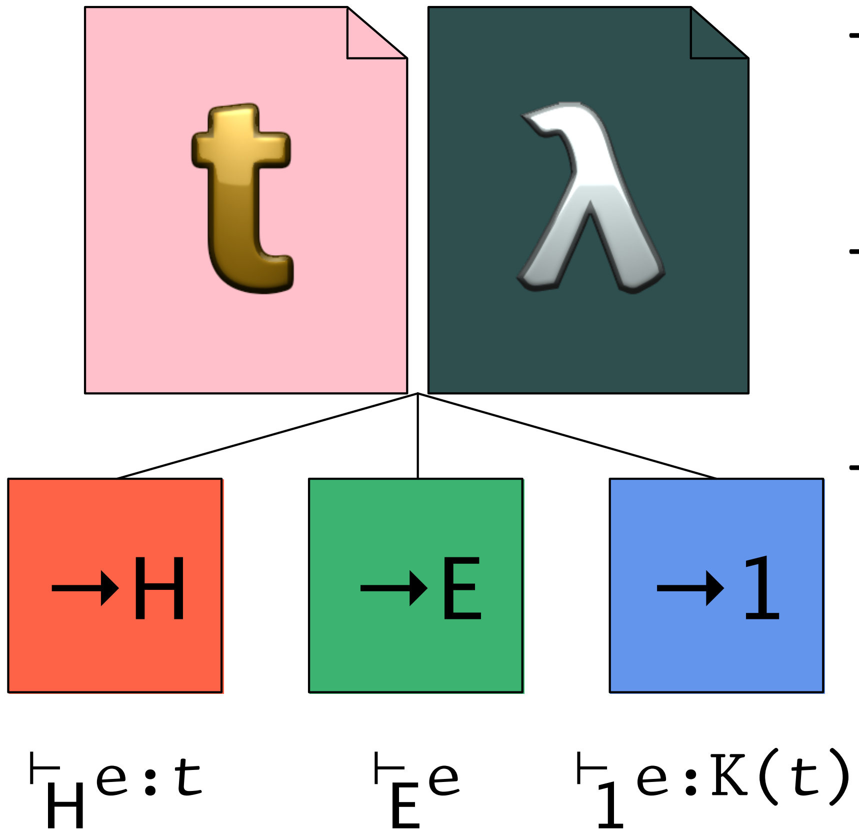


Same type soundness?



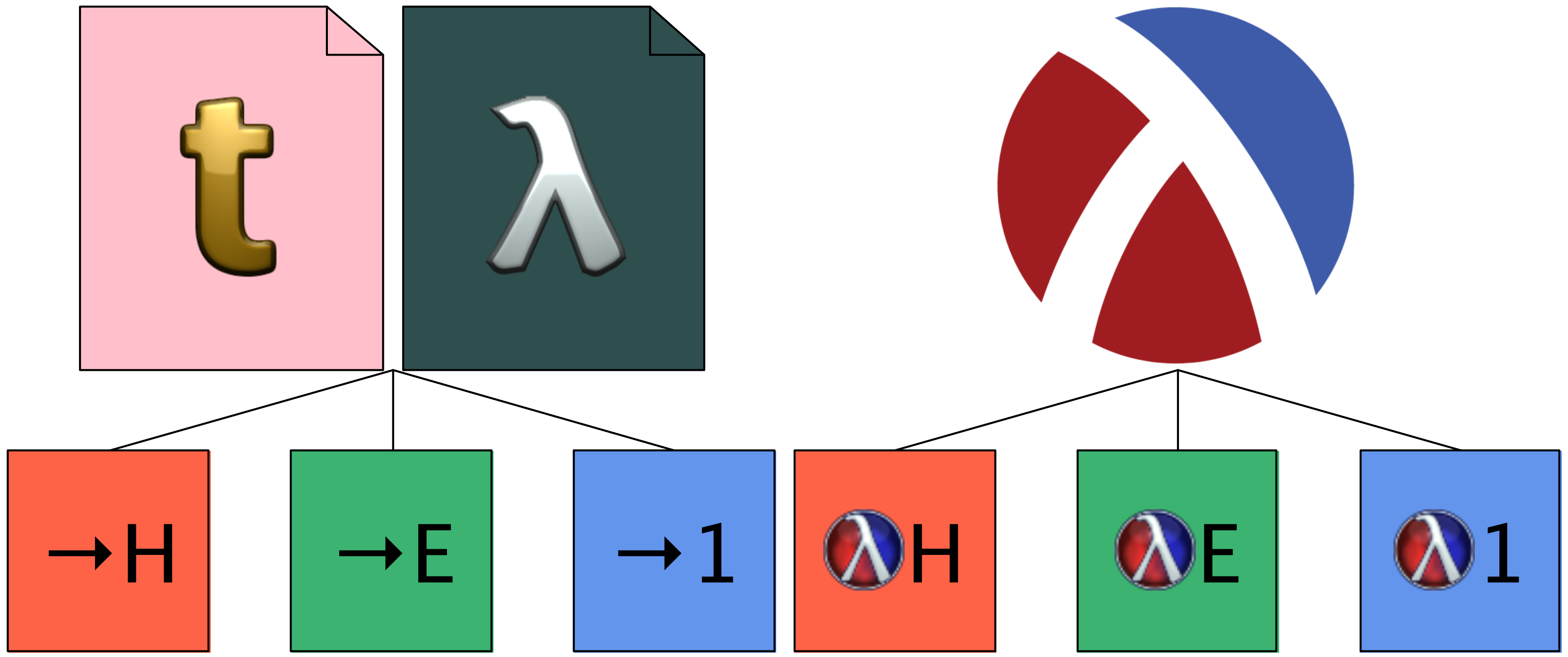
Same type soundness?

No!

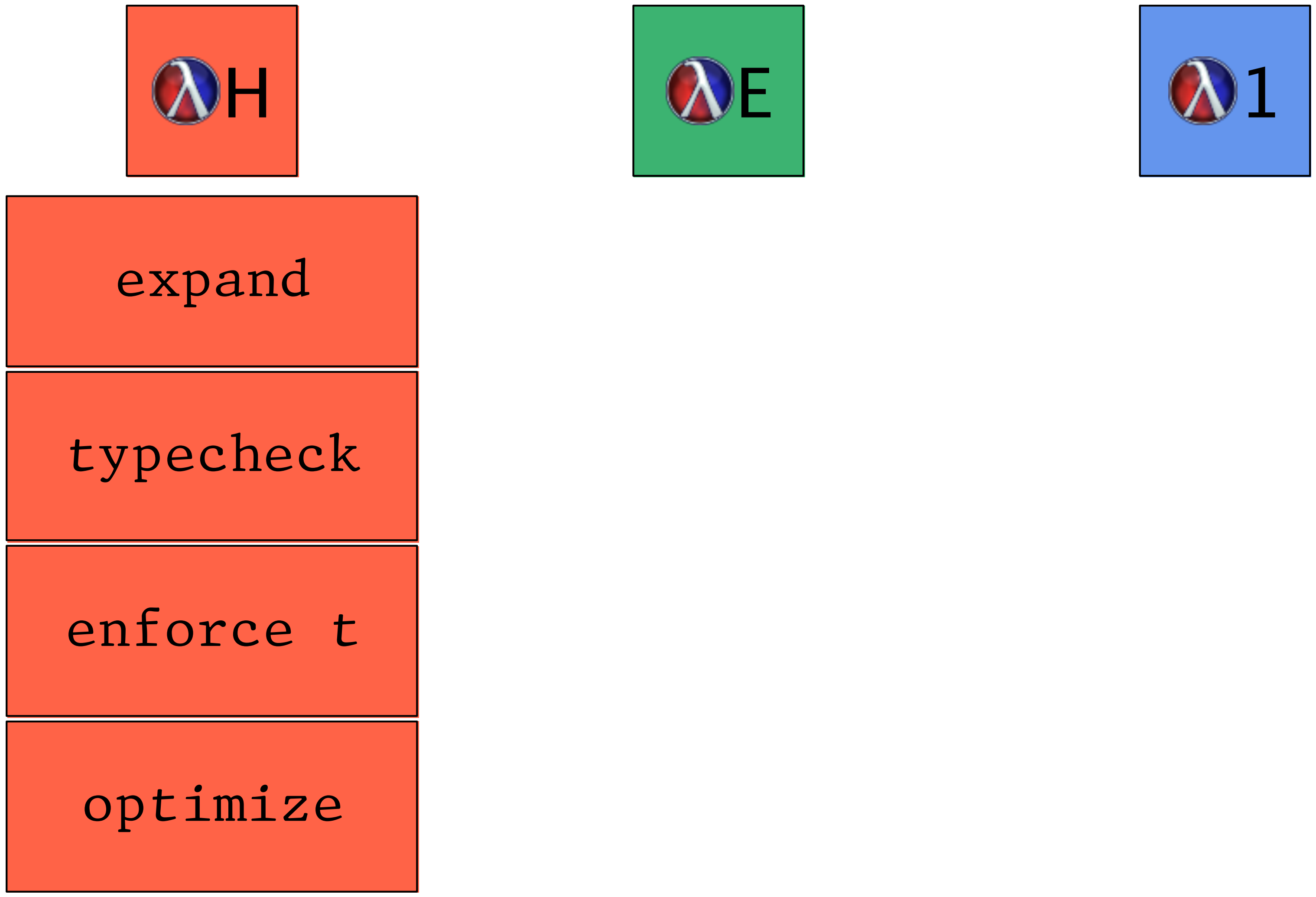


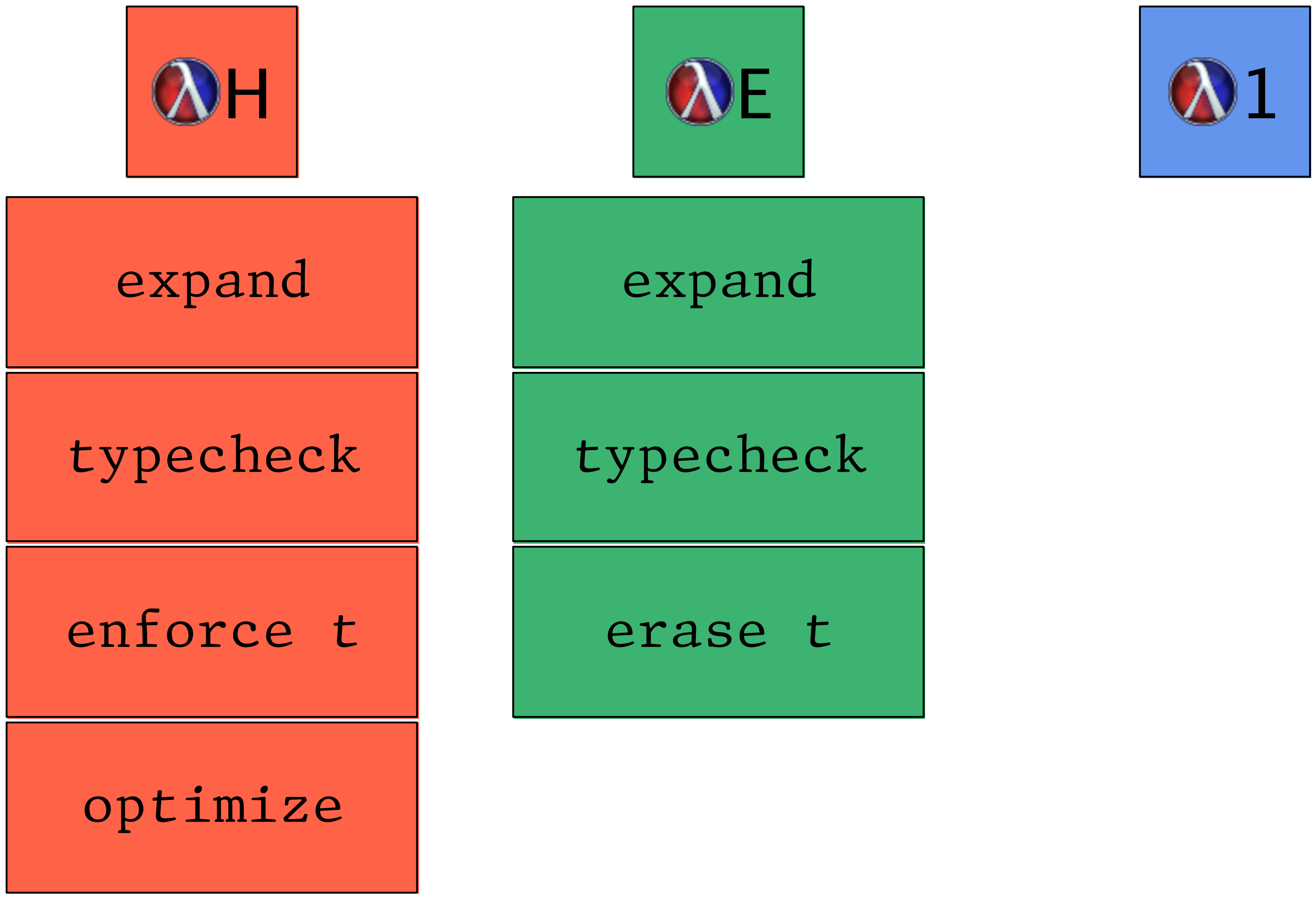
- $\vdash_H e : t$ sound for $\to H$
- $\vdash_E e$ sound for $\to E$
- $\vdash_1 e : K(t)$ sound for $\to 1$

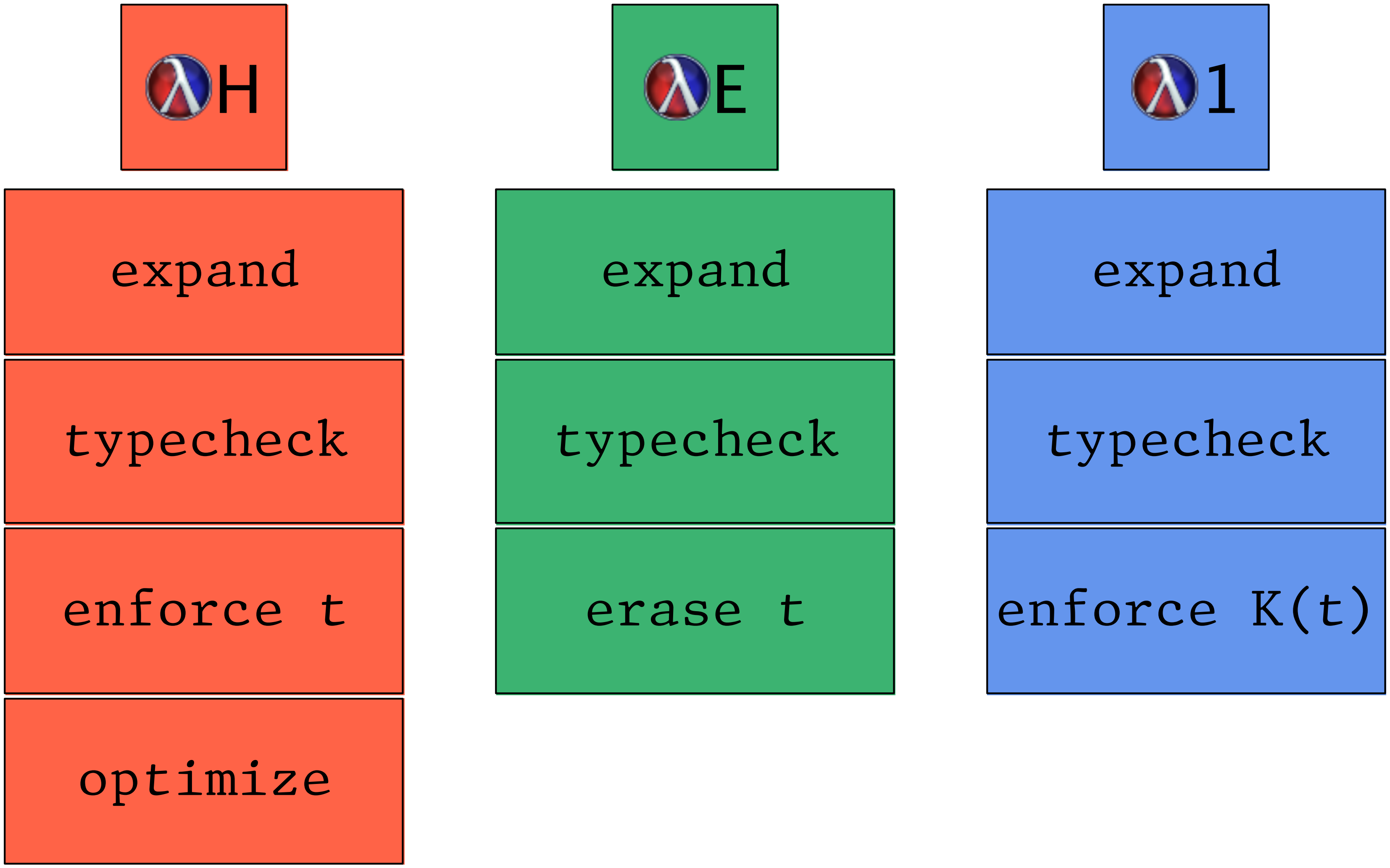
Implementation

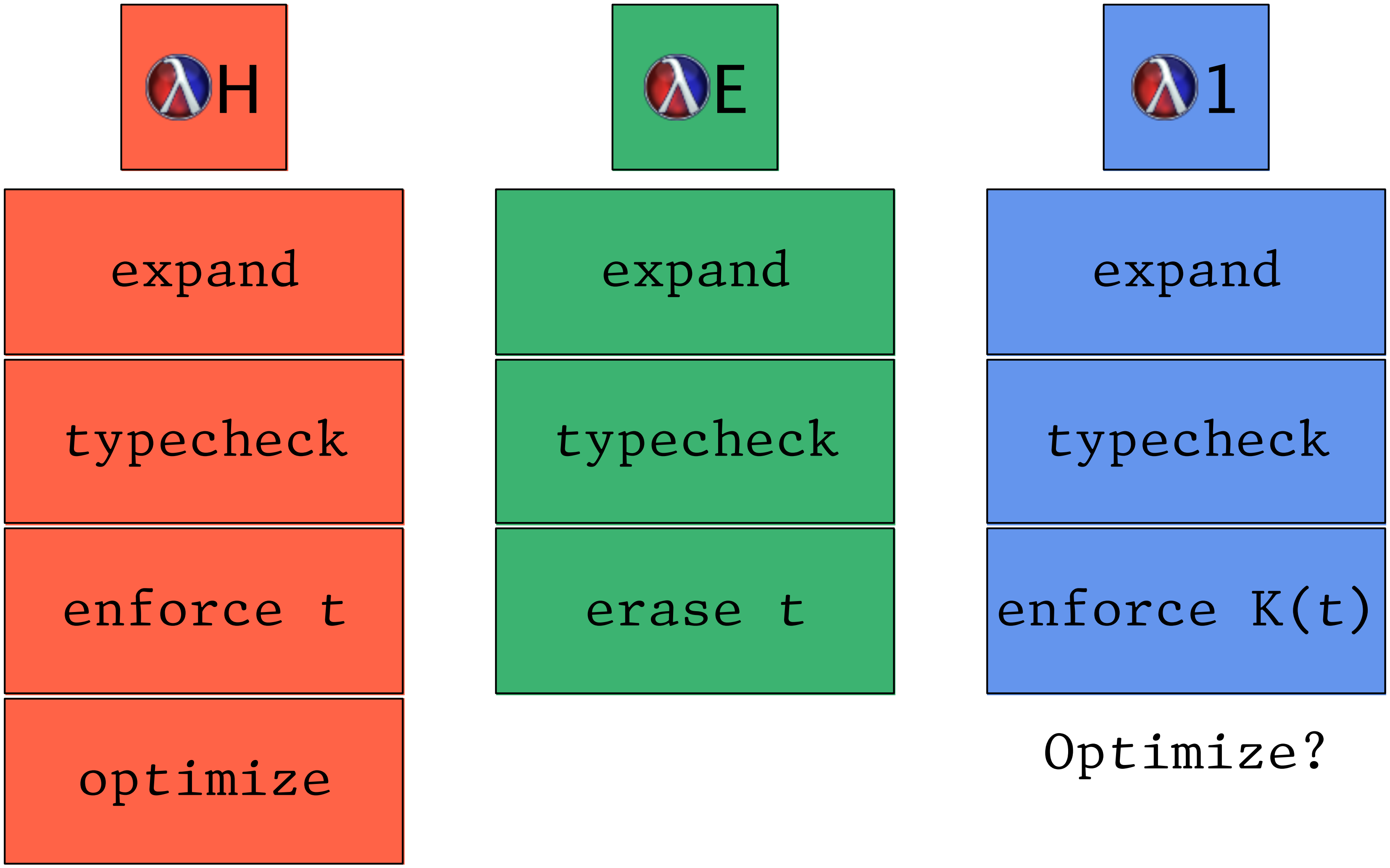

→H
→E
→1
H
E
1

H
E
1


H
E
1
expand
typecheck
enforce t
optimize


H
expand
typecheck
enforce t
optimize
E
expand
typecheck
erase t
1


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


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

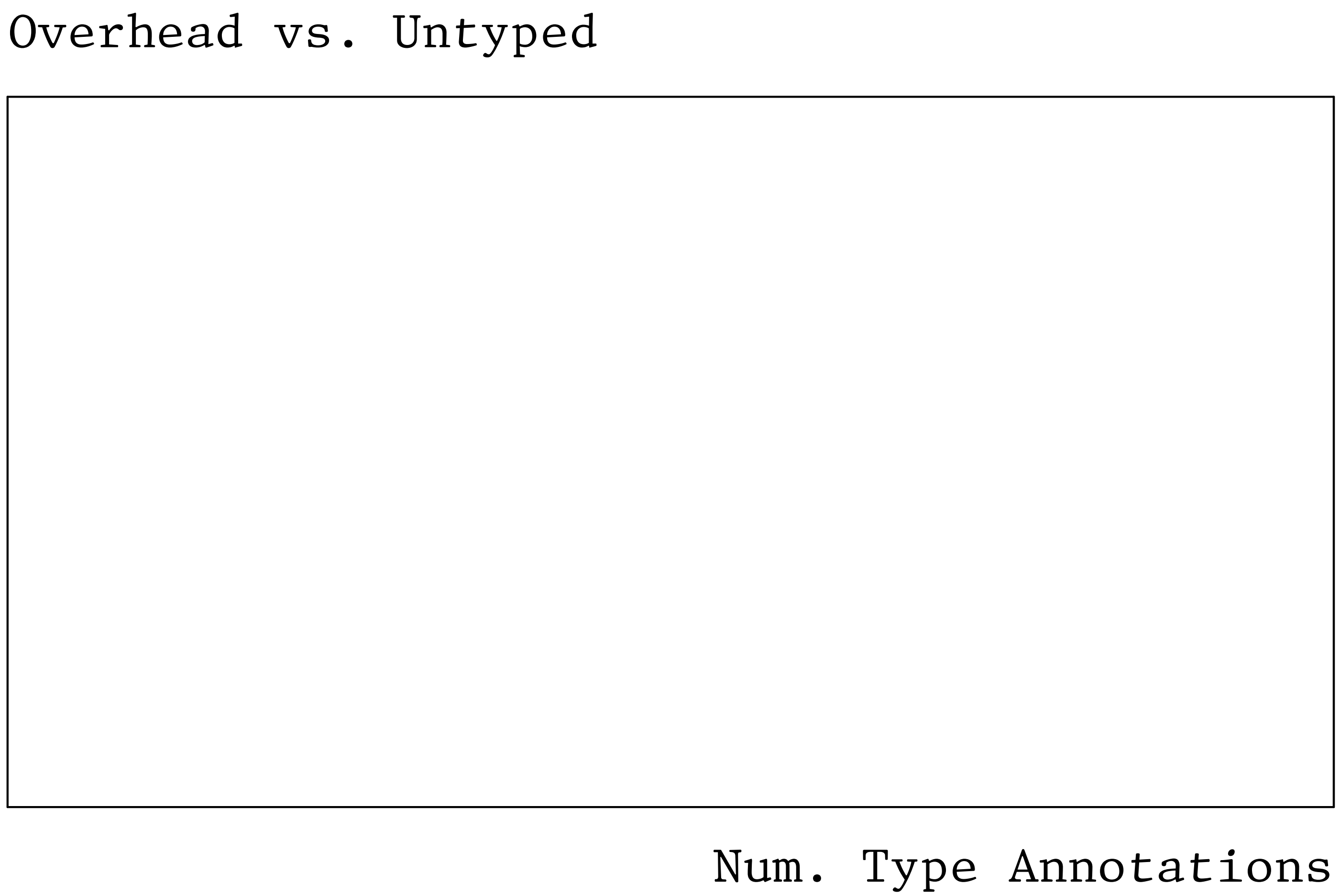
Experiment

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

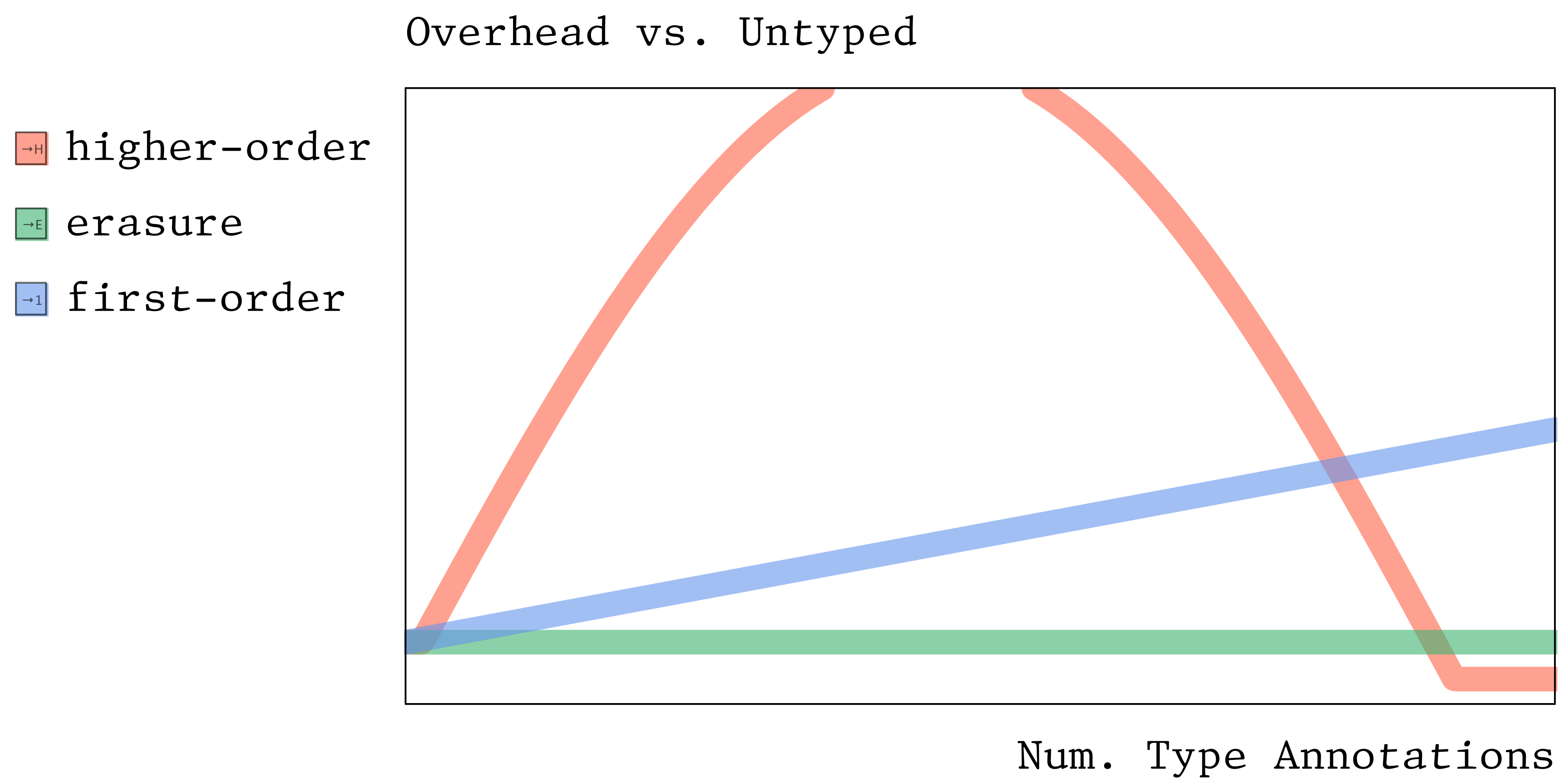
docs.racket-lang.org/gtp-benchmarks

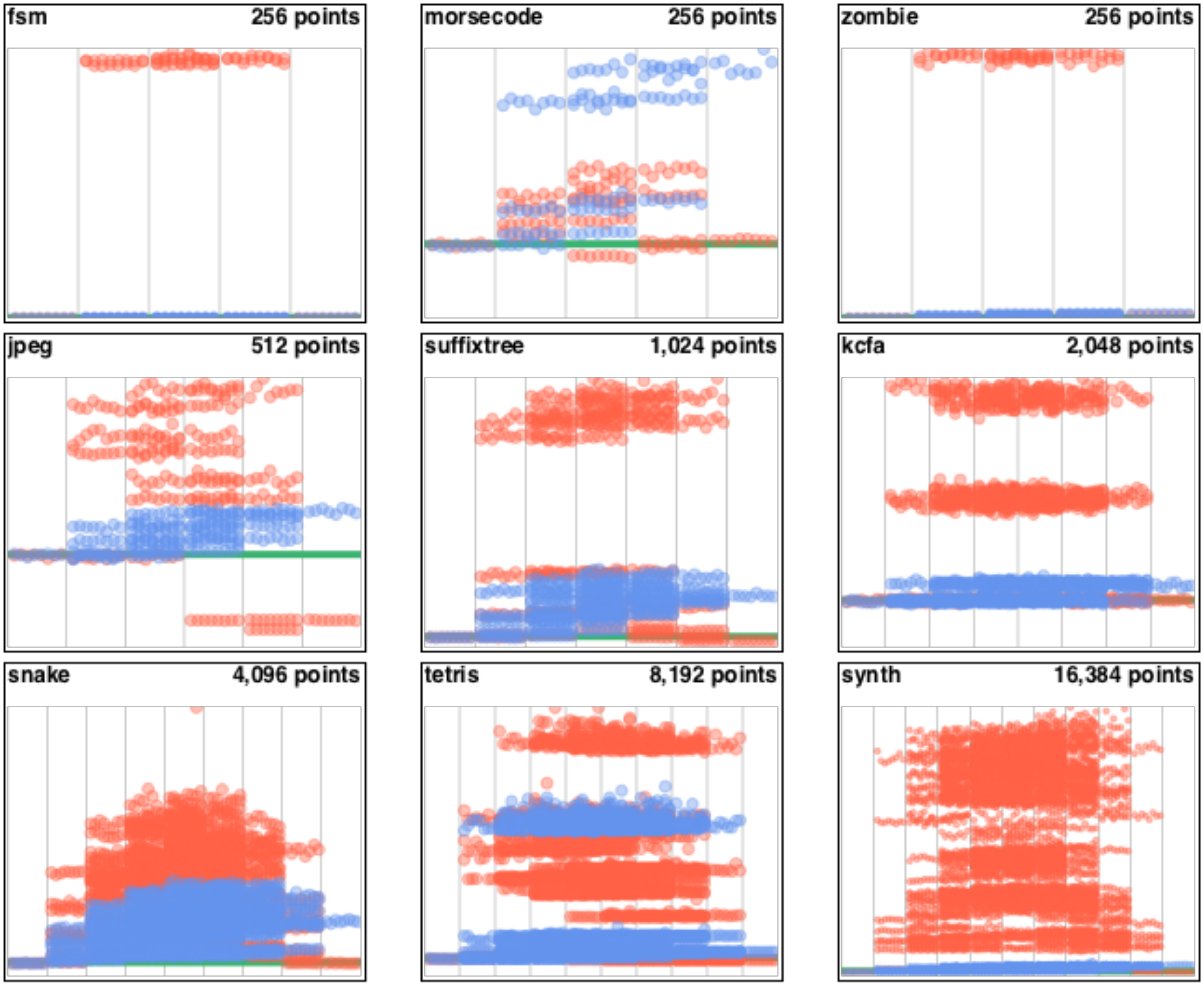
Results

Typical program



Typical program




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

Implications

Theory Implications

$$\boxed{\to H} \supset \boxed{\to 1} \supset \boxed{\to E}$$

$\to H$ `progress + preserves types`

$\to E$ `progress`

$\to 1$ `progress + preserves type constructors`

Theory Implications

$$\boxed{\rightarrow\mathrm{H}} \supset \boxed{\rightarrow 1} \supset \boxed{\rightarrow\mathrm{E}}$$

$\rightarrow$H progress + preserves types

$\rightarrow$E progress

$\rightarrow$1 progress + preserves type constructors

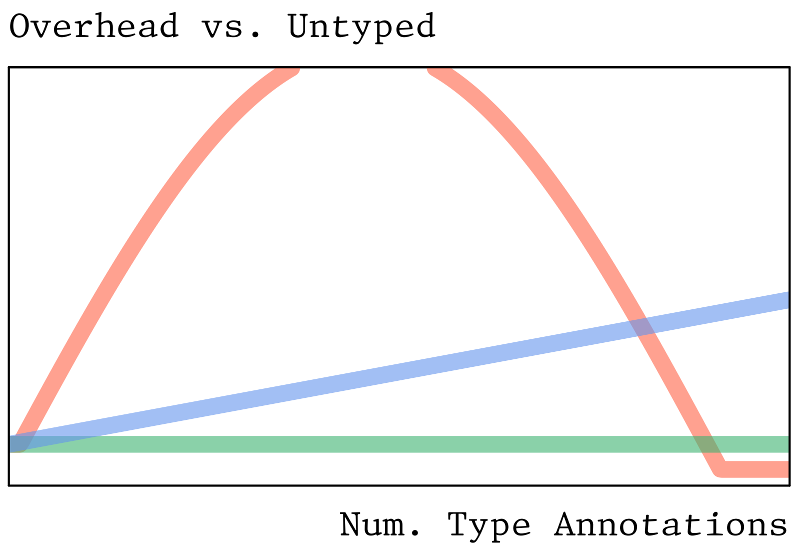
- $\rightarrow$H refines $\rightarrow$1
- $\rightarrow$1 refines $\rightarrow$E

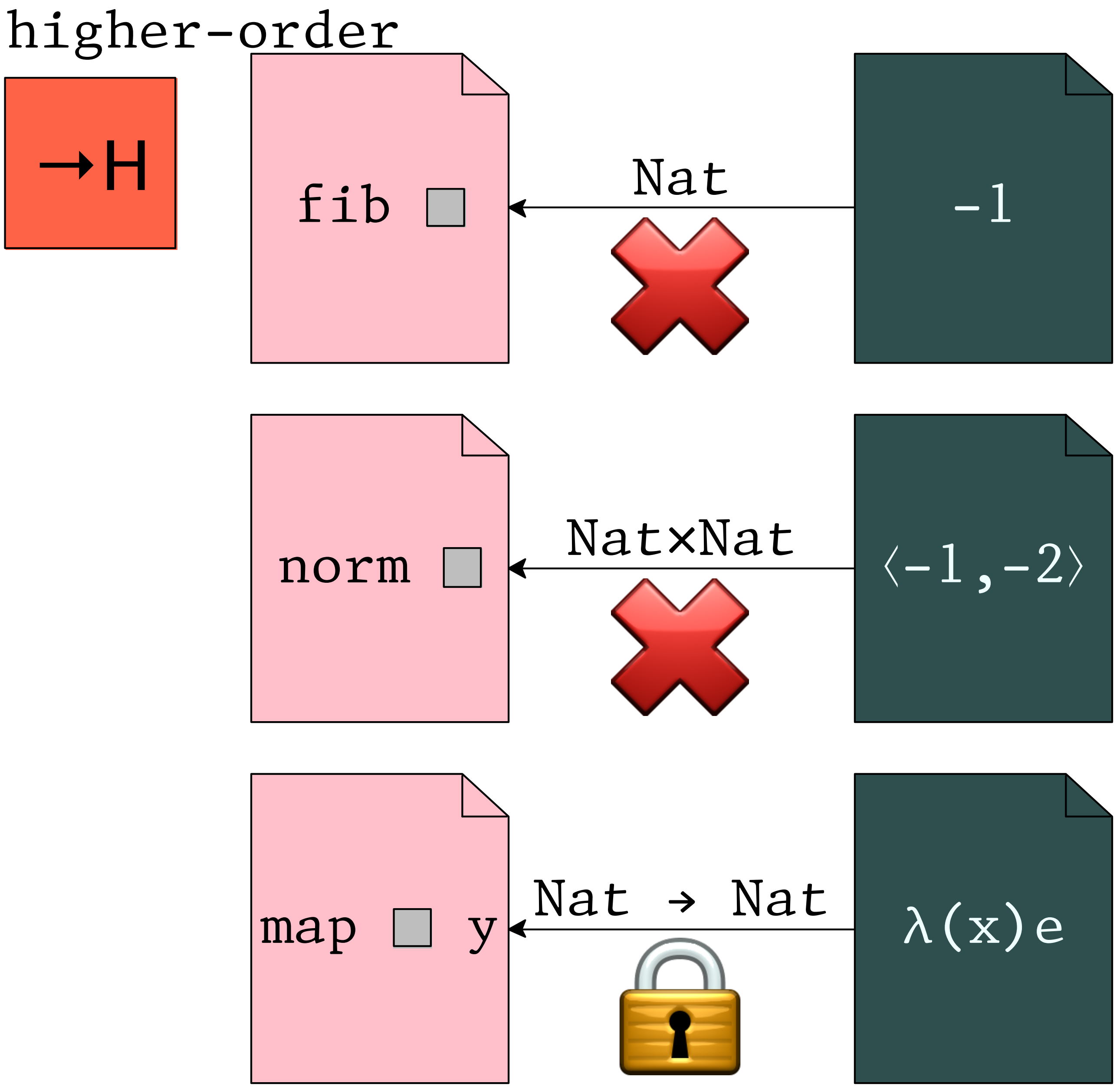
Performance Implications

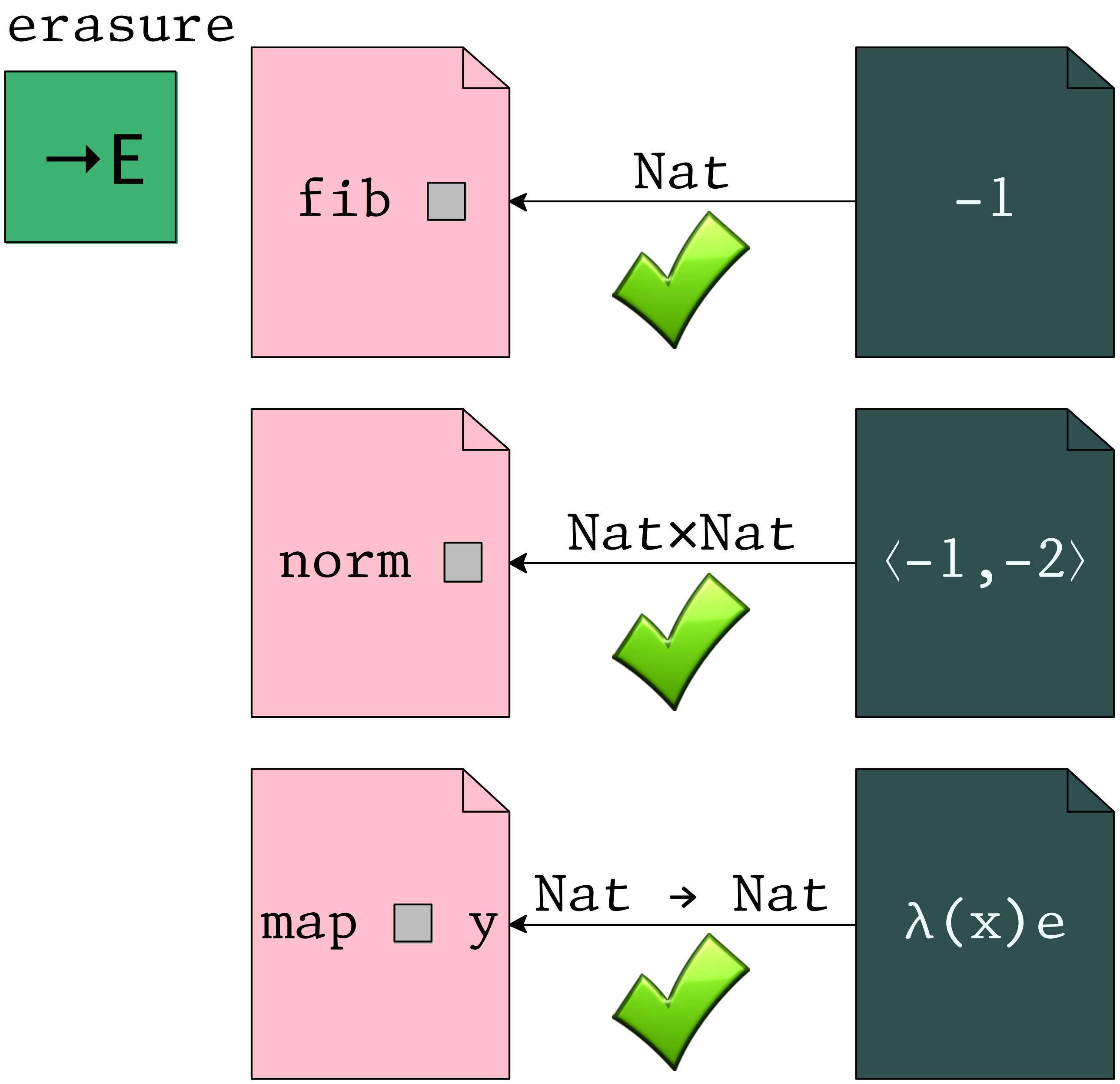
→H add types to 'packages'

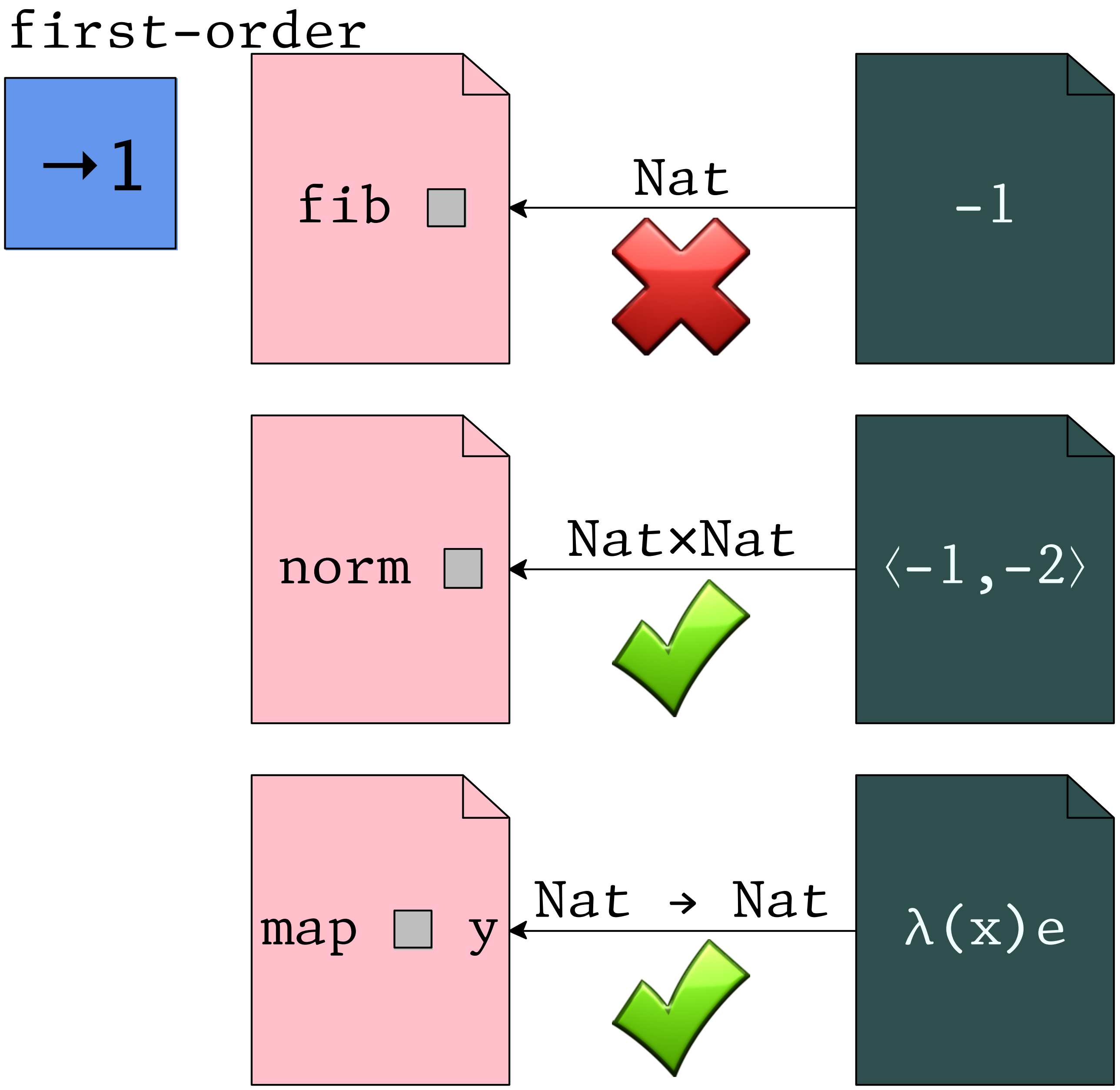
→E add types anywhere

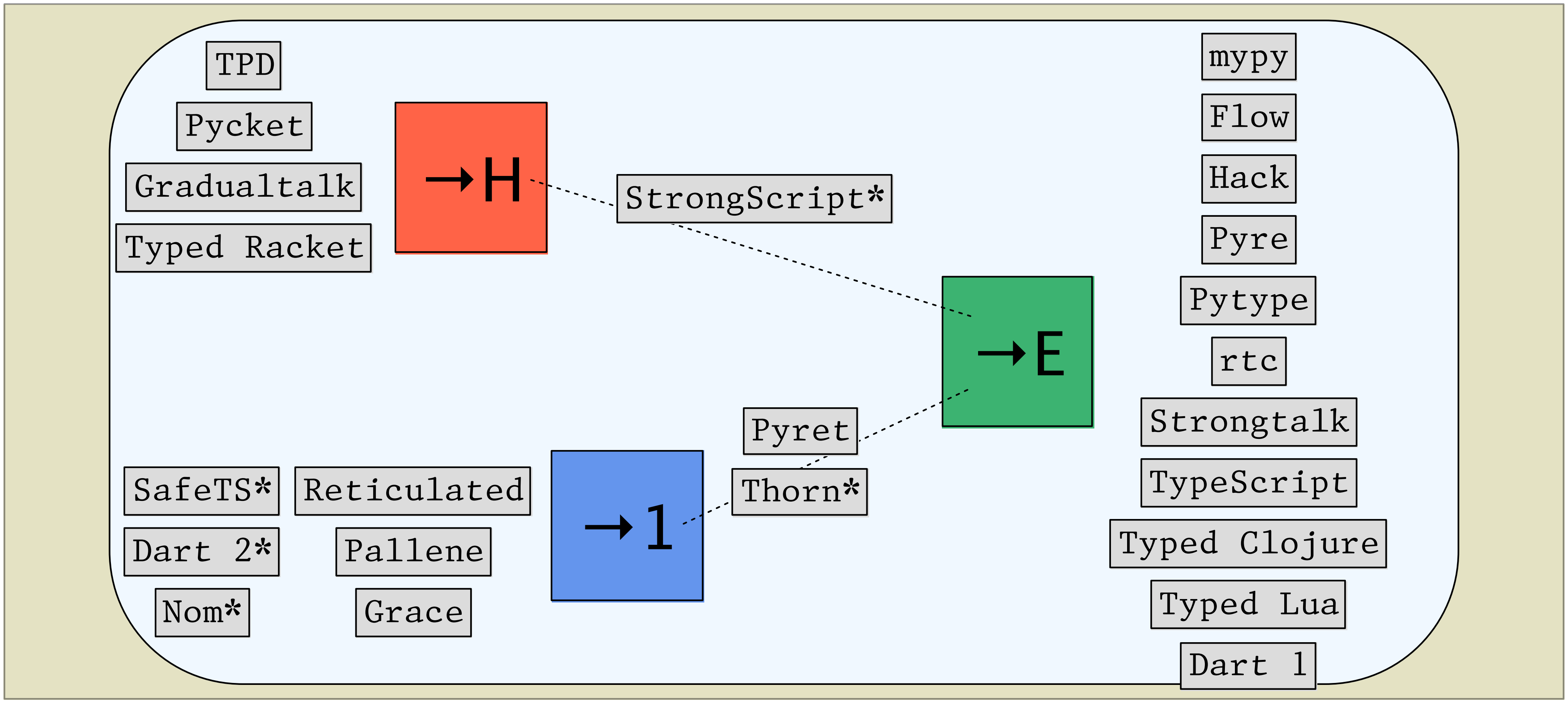
→1 add types sparingly

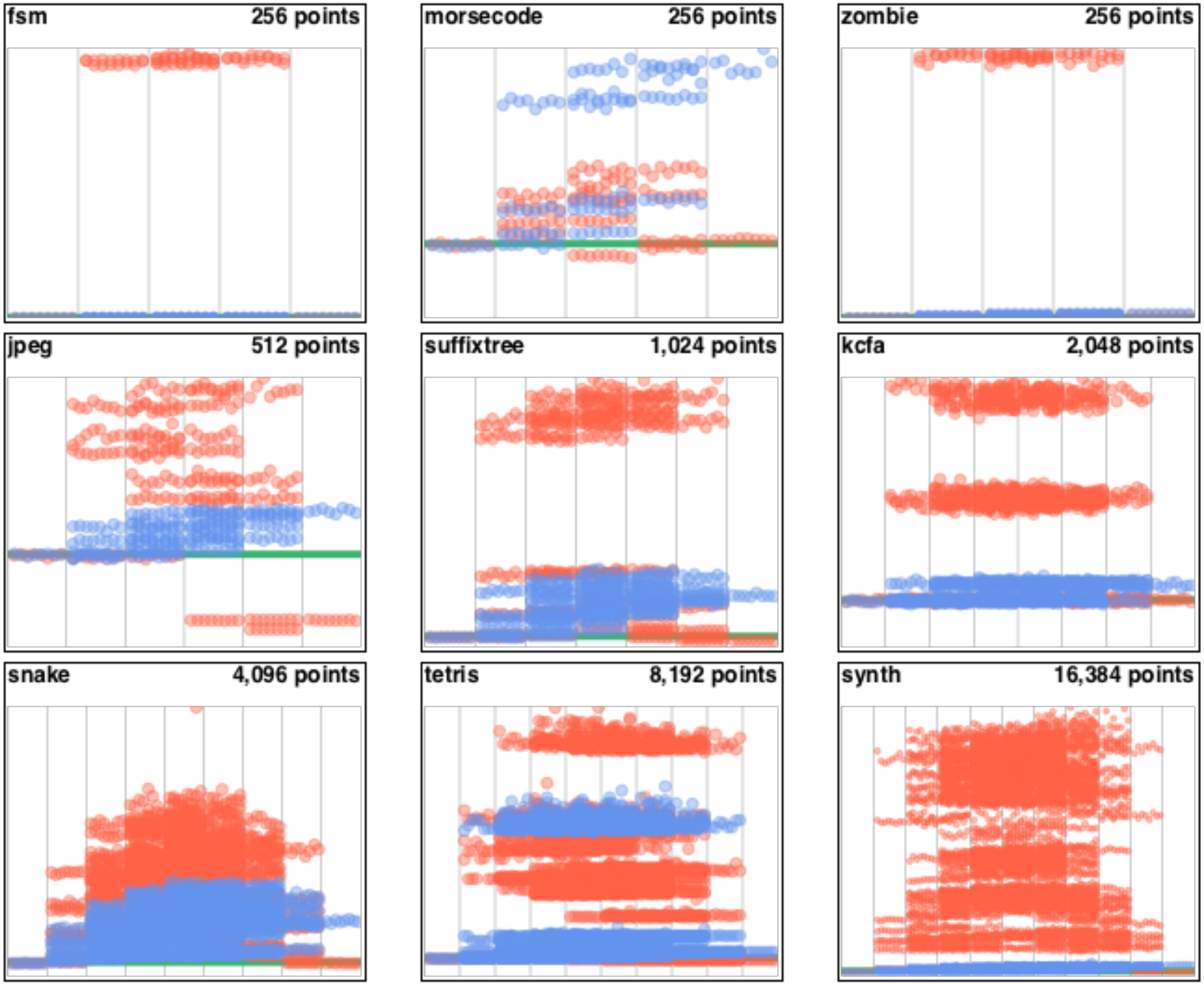



higher-order
→H
fib □
Nat
-1
norm □
Nat×Nat
⟨-1,-2⟩
map □ y
Nat → Nat
λ(x)e


erasure
→E
fib □
Nat
-1
norm □
Nat×Nat
⟨-1,-2⟩
map □ y
Nat → Nat
λ(x)e


first-order
→1
fib □
Nat
-1
norm □
Nat×Nat
⟨-1,-2⟩
map □ y
Nat → Nat
λ(x)e


TPD
Pycket
Gradualtalk
Typed Racket
→H
StrongScript*
mypy
Flow
Hack
Pyre
Pytype
rtc
Strongtalk
TypeScript
Typed Clojure
Typed Lua
Dart 1
→E
Pyret
Thorn*
→1
SafeTS*
Dart 2*
Nom*
Reticulated
Pallene
Grace


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

Is type soundness all-or-nothing?

Can adding types slow down a program?

Is type soundness all-or-nothing?

What invariants should the language guarantee?

Can adding types slow down a program?

Yes, through interaction with untyped code (or data)