Types for (In)correctness

TRIPLE 2026

Photo by Cristina Aranda, (pexels.com)

PLRG

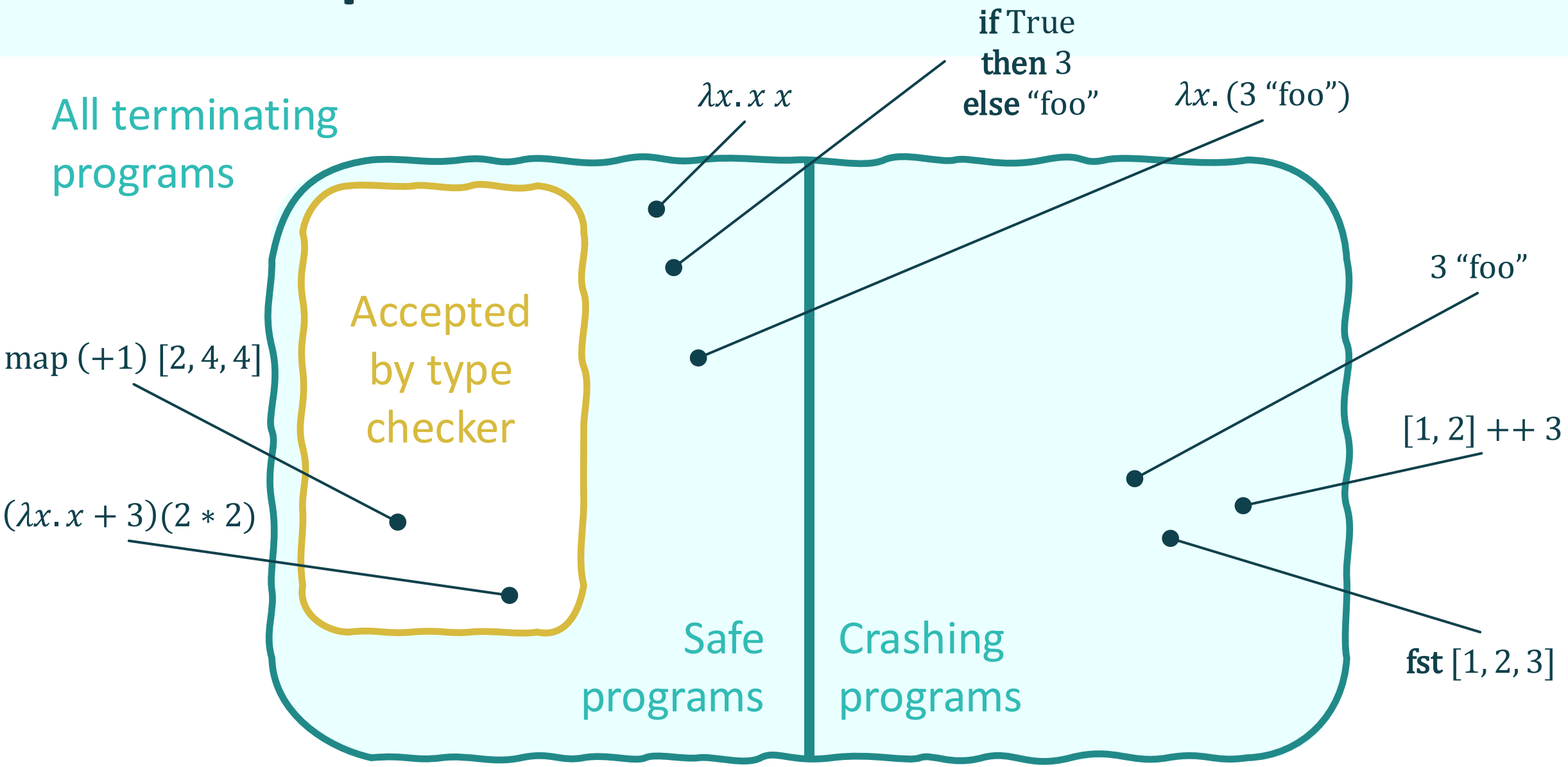
Steven Ramsay
Programming Languages Research Group
University of Bristol, UK

Well-Typed Programs Don’t Go Wrong

Most type systems for programming languages satisfy (in theory):

> Every program accepted by the type checker either diverges or evaluates to a value.

Landscape


All terminating programs
Accepted by type checker
Safe programs
Crashing programs
λx. x x
if True then 3 else "foo"
λx. (3 "foo")
map (+1) [2, 4, 4]
(λx. x + 3)(2 * 2)
3 "foo"
[1, 2] ++ 3
fst [1, 2, 3]

Mathematics of Type Systems

Type Assignment

$3 + 2 : Int$

$3 + x : Int$

$map\ (+1)\ [2, 3] : Int$

$\lambda x. 3 + x : Int \to Int$

Typing Judgement

Term

Assumptions

$$x{:}\,B_1, \ldots, x{:}\,B_n \vdash M : A$$

Type

$$\vdash 3 + 2 : Int$$

$$x{:}\,Int \vdash 3 + x : Int$$

$$f{:}\,\mathrm{Int} \to \mathrm{Int} \vdash map\ f\ [2,3] : [\mathrm{Int}]$$

$$x{:}\,[Int] \vdash 3 + x : Int$$

Core Typing Rules

Γ An arbitrary set of assumptions on the types of variables

$$\frac{}{\Gamma, x : A \vdash x : A}\ (\text{Var})$$

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B}\ (\text{App})$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.\, M : A \to B}\ (\text{Abs})$$

Recalling an assumption

$$\frac{}{\Gamma, x : A \vdash x : A}\ (\text{Var})$$

$$f : \text{Int} \to \text{String},\ \ z : \text{Int} \vdash f\ : \text{Int}\ \to \text{String}$$

$$f : \text{Int} \to \text{String},\ \ z : \text{Int} \vdash z : \text{Int}$$

$$f : \text{Int} \to \text{String},\ \ z : \text{Int} \nvdash z : \text{Int} \to \mathit{String}$$

Applying a function to an input

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B}\ (\text{App})$$

$$\frac{f : \text{Int} \to \text{String},\ z : \text{Int} \vdash f : \text{Int} \to \text{String} \qquad f : \text{Int} \to \text{String},\ z : \text{Int} \vdash z : \text{Int}}{f : \text{Int} \to \text{String},\ z : \text{Int} \vdash f\ z : \text{String}}$$

Constructing a function

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.\, M : A \to B}\ (\text{Abs})$$

$$\frac{f : \text{Int} \to \text{String},\ \ z : \text{Int} \vdash f\ z : \text{String}}{z : \text{Int} \vdash \lambda f.\, f\ z : (\text{Int} \to \text{String}) \to \text{String}}$$

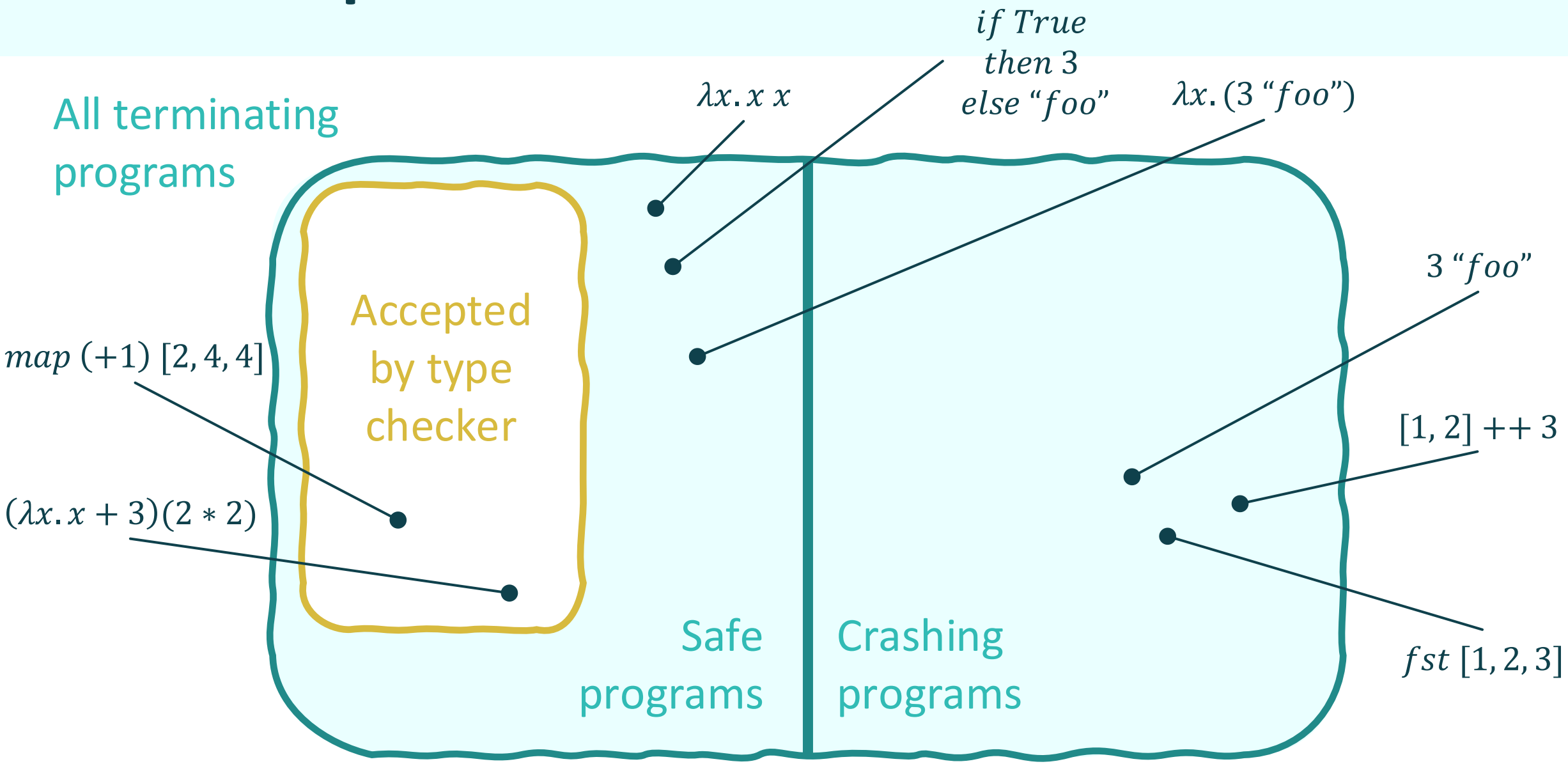
Typing primitives

+ typing rules for all the primitives of the programming language, e.g.

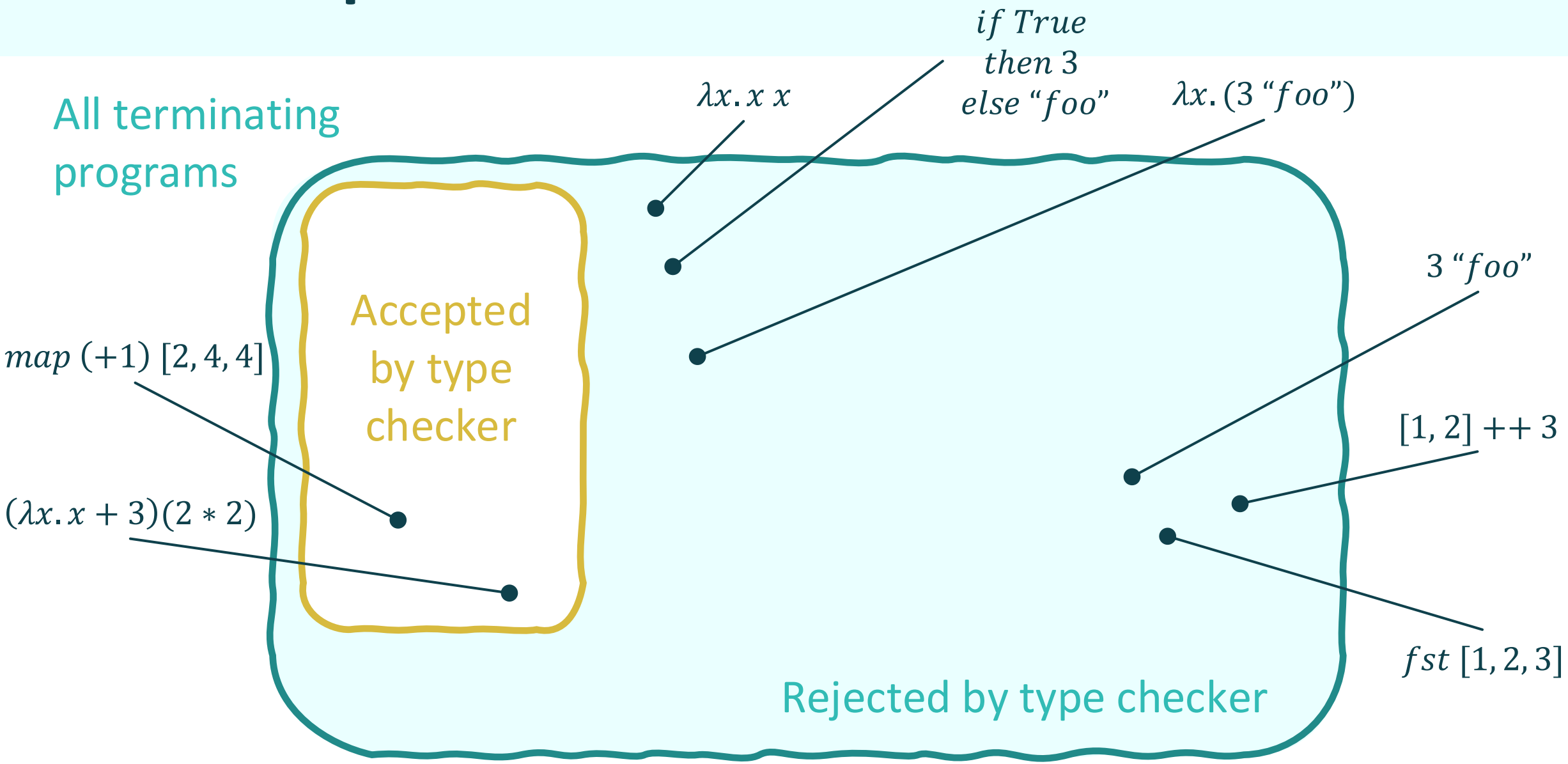
$$\frac{\Gamma \vdash M : \mathrm{Bool} \qquad \Gamma \vdash N : A \qquad \Gamma \vdash P : A}{\Gamma \vdash \textbf{if } M \textbf{ then } N \textbf{ else } P : A}\ (\mathrm{If})$$

$$\not\vdash \textbf{if } \mathrm{True} \textbf{ then } 3 \textbf{ else } \text{“foo”} : A$$

Landscape

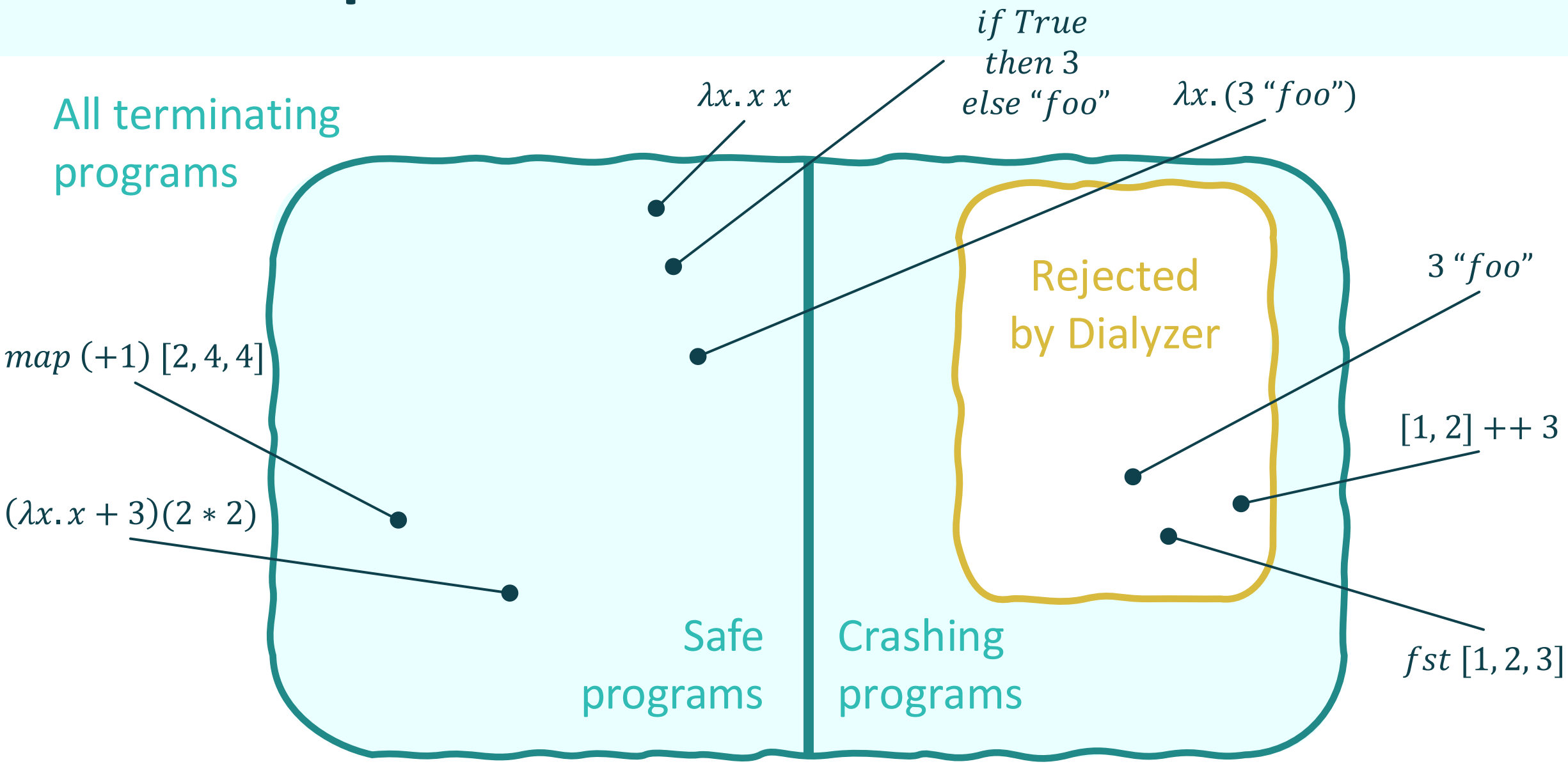

All terminating programs
Accepted by type checker
Safe programs
Crashing programs
$\lambda x.\, x\ x$
$if\ True\ then\ 3\ else\ "foo"$
$\lambda x.\, (3\ "foo")$
$map\ (+1)\ [2, 4, 4]$
$(\lambda x.\, x + 3)(2 * 2)$
$3\ "foo"$
$[1, 2]\ ++\ 3$
$fst\ [1, 2, 3]$

Landscape

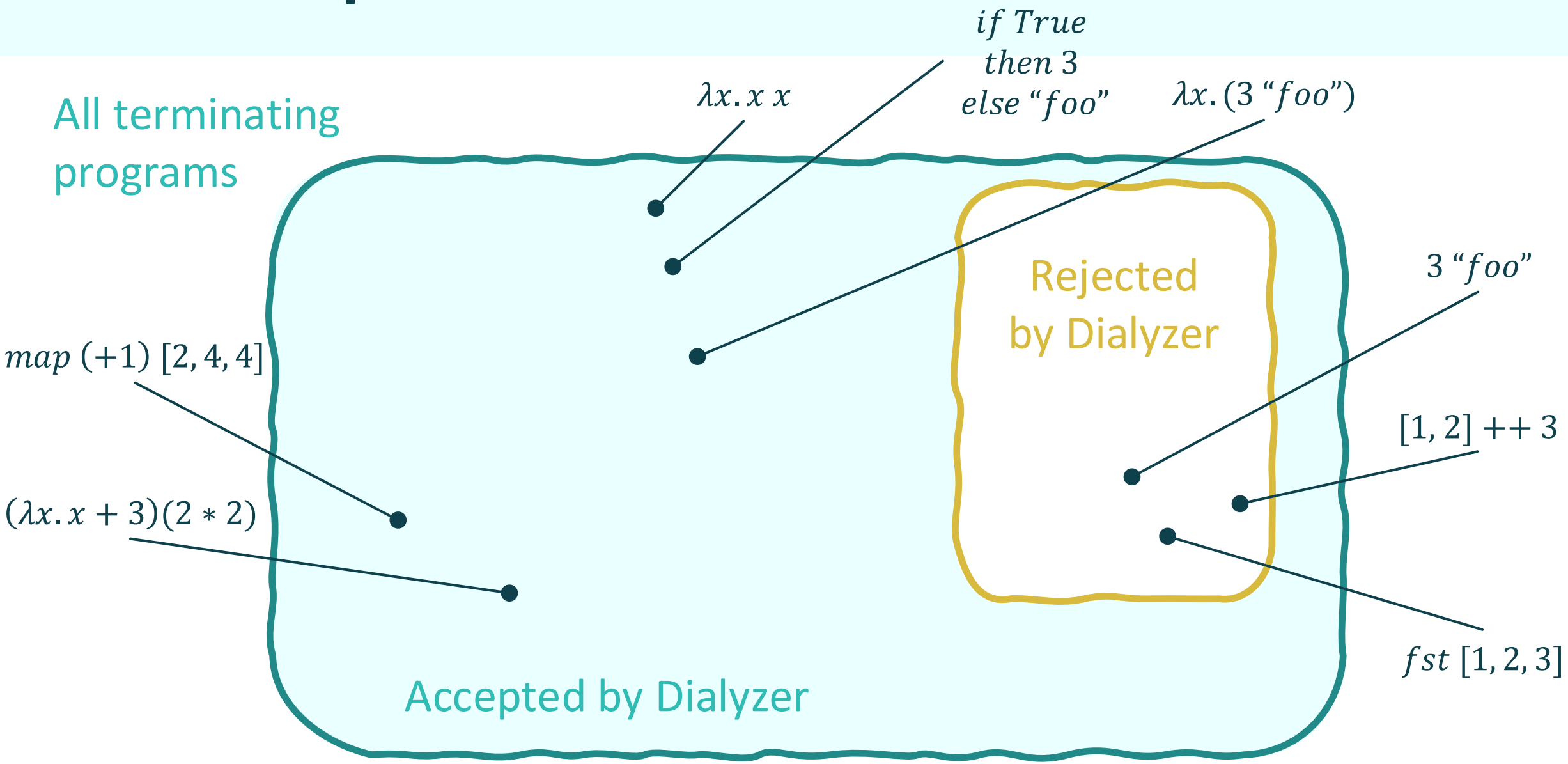

All terminating programs
Accepted by type checker
Rejected by type checker
λx. x x
if True then 3 else "foo"
λx. (3 "foo")
3 "foo"
[1, 2] ++ 3
fst [1, 2, 3]
map (+1) [2, 4, 4]
(λx. x + 3)(2 * 2)

Typing Broken Programs

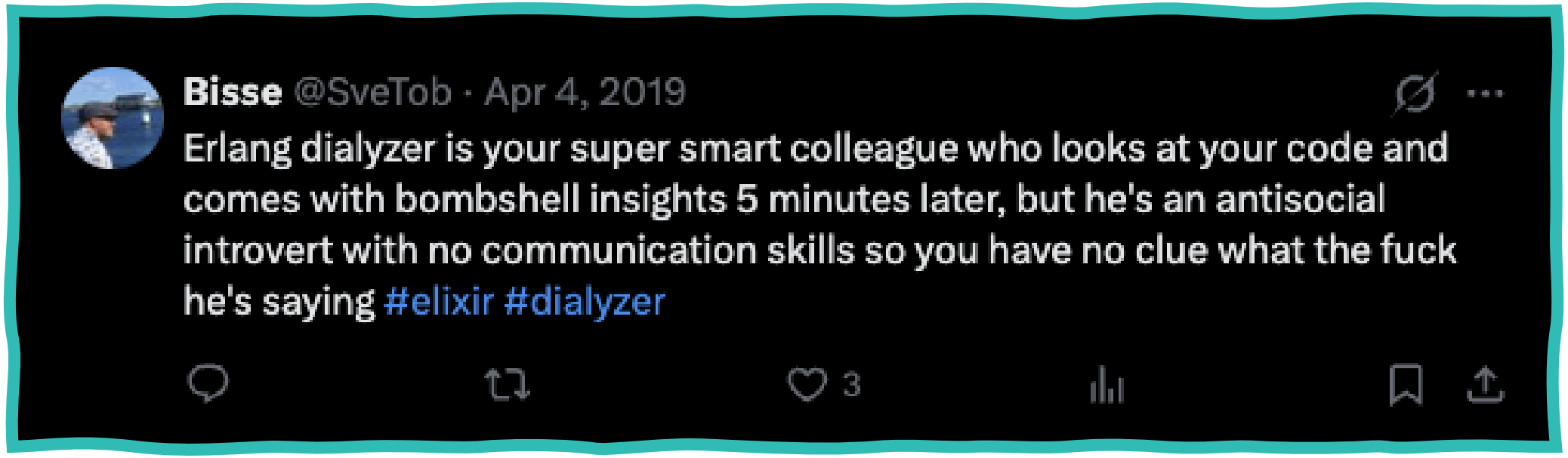
Landscape


All terminating programs
$\lambda x.\, x\ x$
$if\ True\ then\ 3\ else\ "foo"$
$\lambda x.\, (3\ "foo")$
Rejected by Dialyzer
$3\ "foo"$
$map\ (+1)\ [2, 4, 4]$
$[1, 2]\ ++\ 3$
$(\lambda x.\, x + 3)(2 * 2)$
$fst\ [1, 2, 3]$
Safe programs
Crashing programs

Landscape


All terminating programs
$\lambda x.\, x\ x$
$if\ True\ then\ 3\ else\ "foo"$
$\lambda x.\, (3\ "foo")$
Rejected by Dialyzer
$3\ "foo"$
$map\ (+1)\ [2, 4, 4]$
$[1, 2]\ ++\ 3$
$(\lambda x.\, x + 3)(2 * 2)$
$fst\ [1, 2, 3]$
Accepted by Dialyzer

Dialyzer



(Tweet taken from Stavros Aronis' Code BEAM '19 talk.)

Success Types

```
-type face() ::
  1 .. 9 | king | queen | jack | joker.

value(F) ->
  case F of
    jack -> 10;
    queen -> 11;
    king -> 12;
    N -> N
  end.
```

All the following success typings are accepted:

$$value : [1 \,..\, 9] \cup king \cup queen \cup jack \rightarrow int$$

$$value : king \cup queen \cup jack \rightarrow 10 \,..\, 12$$

$$value : king \cup queen \cup jack \rightarrow 13 \,..\, 20$$

$$value : string \rightarrow string \rightarrow int$$

Two-Sided Type Systems

Type System as Proof System

Judgements:

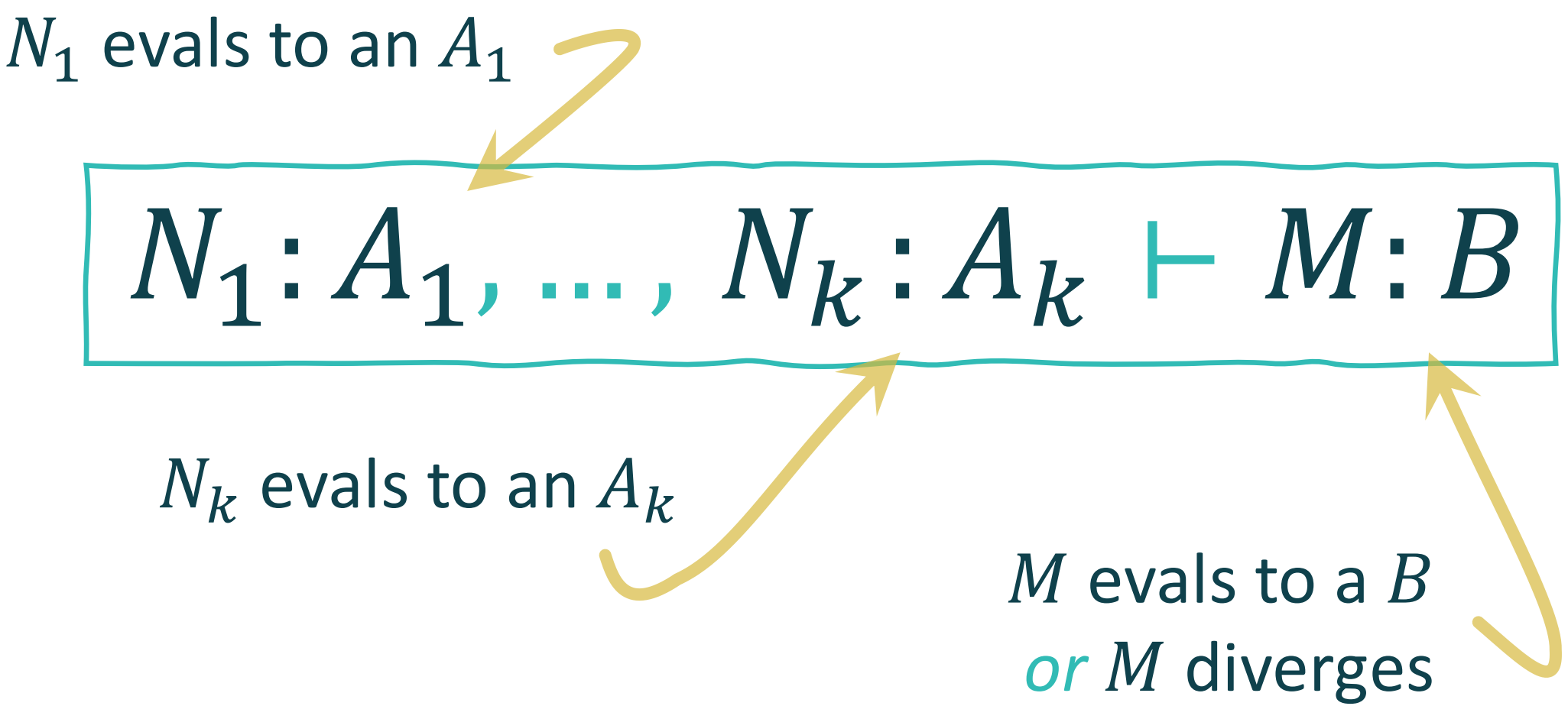
$$x_1 : A_1, \ldots, x_k : A_k \vdash M : B$$

Atomic formulas:

$$M : A$$

Term M restricted to variable when formula is an assumption.

Two-Sided Judgement



PCF Examples

$$(\lambda x.\, x)\ y : \mathsf{Int} \quad \vdash \quad y : \mathsf{Int}$$

$$\textbf{if } z \geq 0 \textbf{ then } M \textbf{ else } N : \mathsf{Int} \quad \vdash z : \mathsf{Int}$$

$$(\lambda x.\, x)\ y : \mathsf{Int}, \quad \textbf{if } z \geq 0 \textbf{ then } M \textbf{ else } N : \mathsf{Int} \quad \vdash \ y + z : \mathsf{Int}$$

Disjunction on the Right

Several formulas on the RHS understood disjunctively:

$$\textbf{if } x \textbf{ then } y \textbf{ else } z : \mathrm{Nat} \quad \vdash y : \mathrm{Nat}, \quad z : \mathrm{Nat}$$

The empty RHS is understood as absurdity:

$$x : \mathrm{Nat} \to \mathrm{Nat}, \quad x + y : \mathrm{Nat} \quad \vdash$$

Sequent Rules

$$\frac{\Gamma \vdash M : \mathrm{Bool}, \Delta \quad \Gamma \vdash N : A, \Delta \quad \Gamma \vdash P : A, \Delta}{\Gamma \vdash \textbf{if } M \textbf{ then } N \textbf{ else } P : A, \Delta}\ (\mathrm{IfR})$$

$$\frac{\Gamma, M : \mathrm{Bool} \vdash \Delta}{\Gamma, \textbf{if } M \textbf{ then } N \textbf{ else } P : A \vdash \Delta}\ (\mathrm{IfL}_1)$$

$$\frac{\Gamma, N : A \vdash \Delta \quad \Gamma, P : A \vdash \Delta}{\Gamma, \textbf{if } M \textbf{ then } N \textbf{ else } P : A \vdash \Delta}\ (\mathrm{IfL}_2)$$

Co-arrow type

$$A \nrightarrow B$$

"*A* coto *B*"

The type of all values that are not functions which necessarily require an A to produce a B

Sufficiency

$$\frac{\Gamma,\ x : A \vdash M : B,\ \Delta}{\Gamma \vdash \lambda x.\, M : A \to B,\ \Delta}\ (\text{AbsR})$$

$\vdash \lambda x.\, 42 : \mathsf{Int} \to \mathsf{Int}$

$\vdash \mathrm{map}\ f : \mathsf{Nil} \to \mathsf{Nil}$

$\nvdash \mathrm{map}\ f : \mathsf{List}(a) \to \mathsf{Nil}$

Necessity

$$\frac{\Gamma,\ M : B \vdash x : A,\ \Delta}{\Gamma, \lambda x.\, M : A \succ\!\!\!\!\prec B \vdash \Delta}\ (\text{AbsL})$$

$\mathsf{hd} : \mathsf{ConsList}(a) \succ\!\!\!\!\prec a \vdash$

$\mathrm{map}\ f : \mathsf{Nil} \succ\!\!\!\!\prec \mathsf{Nil} \vdash$

$\lambda x.\, 42 : \mathsf{Int} \succ\!\!\!\!\prec \mathsf{Int} \nvdash$

Application on the Left

$$\frac{\Gamma, M : A \nrightarrow B \vdash \Delta \qquad \Gamma, N : A \vdash \Delta}{\Gamma, MN : B \vdash \Delta}\ (\text{AppL})$$

To refute that MN is a B:

- Establish that M is a function that necessarily requires an A to produce a B.
- Establish that N is not an A.

Pure Calculus

(Var)

$$\frac{}{\Gamma, x : A \vdash x : A, \Delta}$$

(AbsL)

$$\frac{\Gamma,\ M : B \vdash x : A,\ \Delta}{\Gamma, \lambda x.\, M : A \rightarrowtail B \vdash \Delta}$$

(AbsR)

$$\frac{\Gamma,\ x : A \vdash M : B,\ \Delta}{\Gamma \vdash \lambda x.\, M : A \to B,\ \Delta}$$

(AppL)

$$\frac{\Gamma, M : A \rightarrowtail B \vdash \Delta \quad \Gamma, N : A \vdash \Delta}{\Gamma, M\ N :\ B \vdash \Delta}$$

(AppR)

$$\frac{\Gamma \vdash M : A \to B, \Delta \quad \Gamma \vdash N : A, \Delta}{\Gamma \vdash MN : B, \Delta}$$

Type of Values

Ok

The type of *all* values

$$\text{power (x, y) = } \textbf{if } y == 0 \textbf{ then } 1 \textbf{ else } x * \text{power } (x, y - 1))$$

$$\text{power} : \text{Int} \times \text{Int} \not\vdash \text{Int} \vdash$$

To return an Int, req two Ints

$$\text{power} : \text{Int} \times \text{Int} \not\vdash \text{Ok} \vdash$$

To return a value, req two Ints

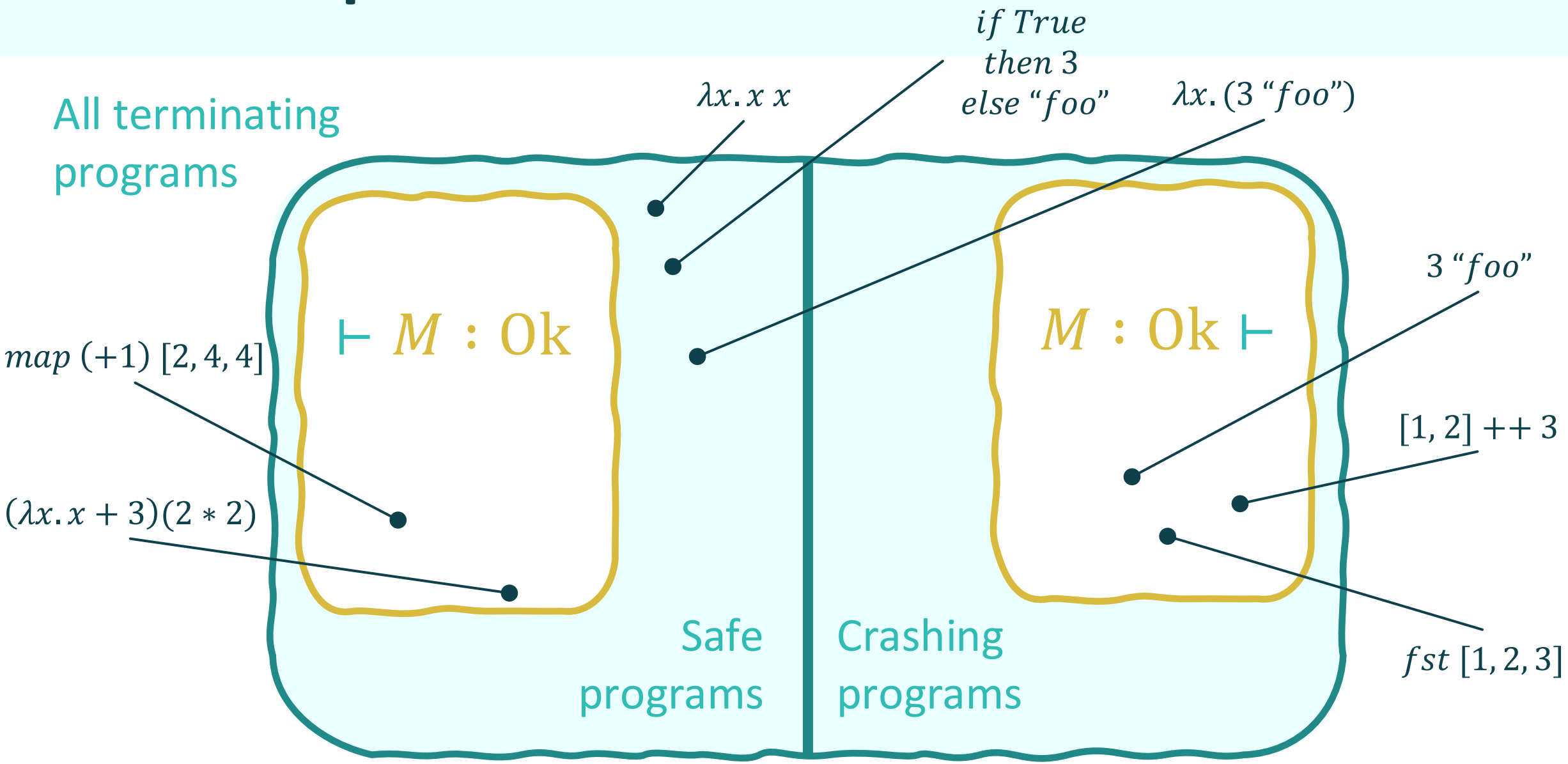
Incorrectness Reasoning

Ok — The type of *all* values

$$\frac{\mathrm{hd} : \mathrm{ConsList}(\mathrm{Ok}) \nrightarrow \mathrm{Ok} \vdash \qquad [] : \mathrm{ConsList}(\mathrm{Ok}) \vdash}{\mathrm{hd}\ [] : \mathrm{Ok} \vdash}$$

“ hd [] *does not* evaluate (reach a value) "

Landscape


All terminating programs
$\lambda x.\, x\ x$
$if\ True\ then\ 3\ else\ "foo"$
$\lambda x.\, (3\ "foo")$
$\vdash M : \mathrm{Ok}$
$M : \mathrm{Ok} \vdash$
$map\ (+1)\ [2,4,4]$
$(\lambda x.\, x + 3)(2 * 2)$
$3\ "foo"$
$[1,2] ++ 3$
$fst\ [1,2,3]$
Safe programs
Crashing programs

Summary

Summary

- Traditional type systems:
  - Certify the good behaviour of programs by proving type assignments.
- Two-sided type systems also:
  - Certify the bad behaviour of programs by refuting type assignments.
- Ongoing research:
  - Algorithms for automatic type inference.
  - Correspondence with certain bi-intuitionistic logics.