Graded Types - Part 4

Grades in the wild via "graded base" and graded monads

Dominic Orchard, 24-28th July 2023, SPLV23

UNIVERSITY OF CAMBRIDGE

University of Kent

Linear logic

$$\Gamma ::= \varnothing \mid \Gamma, x : A$$
$$A ::= A \multimap A'$$

$$\dfrac{}{x : A \vdash x : A}\ \text{ax}$$

$$\dfrac{\Gamma_1, x : A, y : B, \Gamma_2 \vdash t : C}{\Gamma_1, y : B, x : A, \Gamma_2 \vdash t : C}\ \text{ex}$$

$$\dfrac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x\,.\,t : A \multimap B}\ \multimap_i$$

$$\dfrac{\Gamma_1 \vdash t_1 : A \multimap B \qquad \Gamma_2 \vdash t_2 : A}{\Gamma_1, \Gamma_2 \vdash t_1\, t_2 : B}\ \multimap_e$$

Linear logic + !-modality

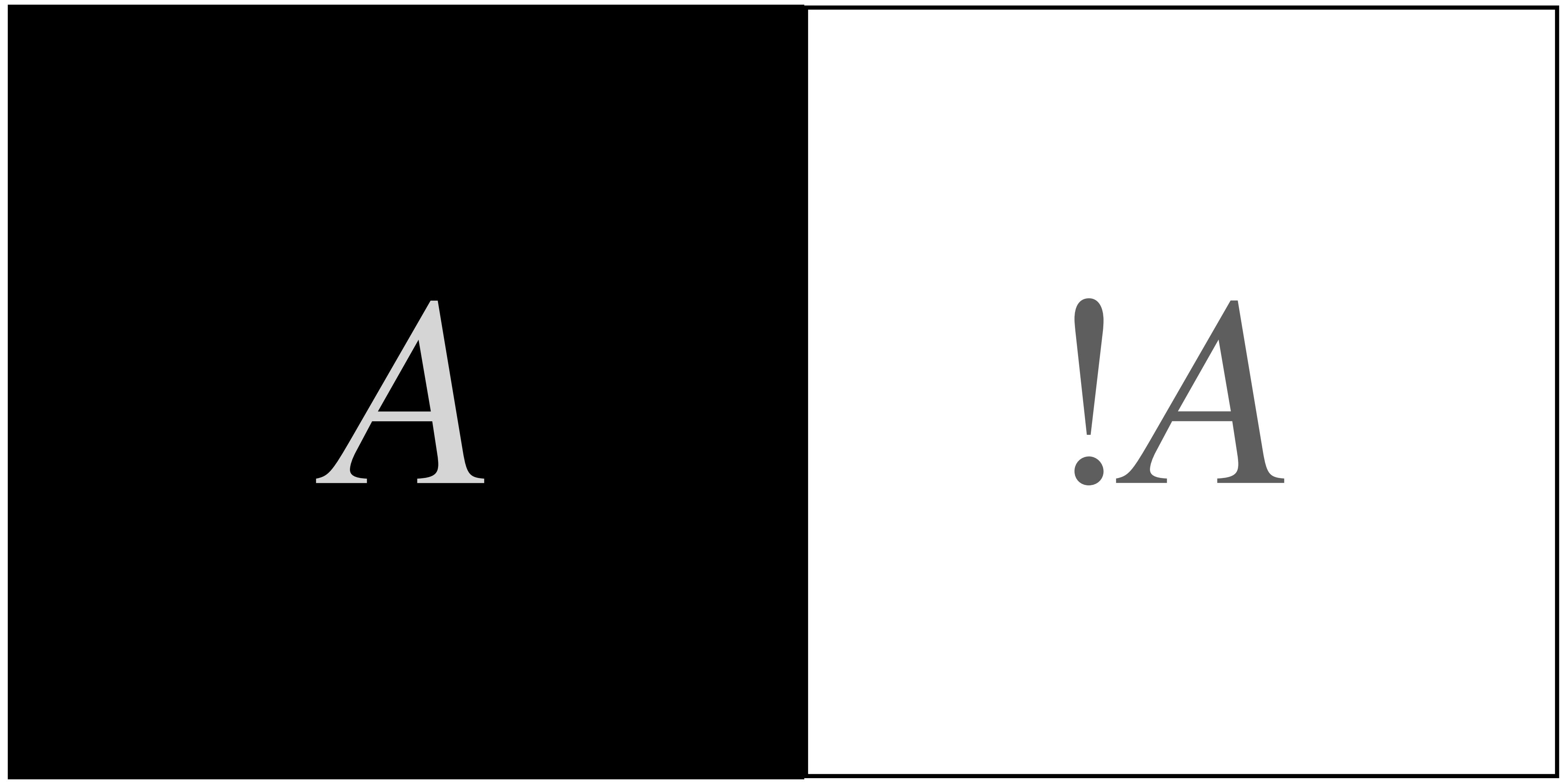
$$\Gamma ::= \varnothing \mid \Gamma, x : A \mid \Gamma, x : [A]$$
$$A ::= A \multimap A' \mid \square A$$

$$\frac{}{x : A \vdash x : A}\ \text{ax} \qquad \frac{\Gamma_1, x : A, y : B, \Gamma_2 \vdash t : C}{\Gamma_1, y : B, x : A, \Gamma_2 \vdash t : C}\ \text{ex} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma, x : [A] \vdash t : B}\ \text{der}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x\,.\,t : A \multimap B}\ \multimap_i \qquad \frac{\Gamma, x : [A], y : [A] \vdash t : B}{\Gamma, z : [A] \vdash t[z/x][z/y] : B}\ \text{contr} \qquad \frac{[\Gamma] \vdash t : B}{[\Gamma] \vdash [t] : \square B}\ \square_i$$

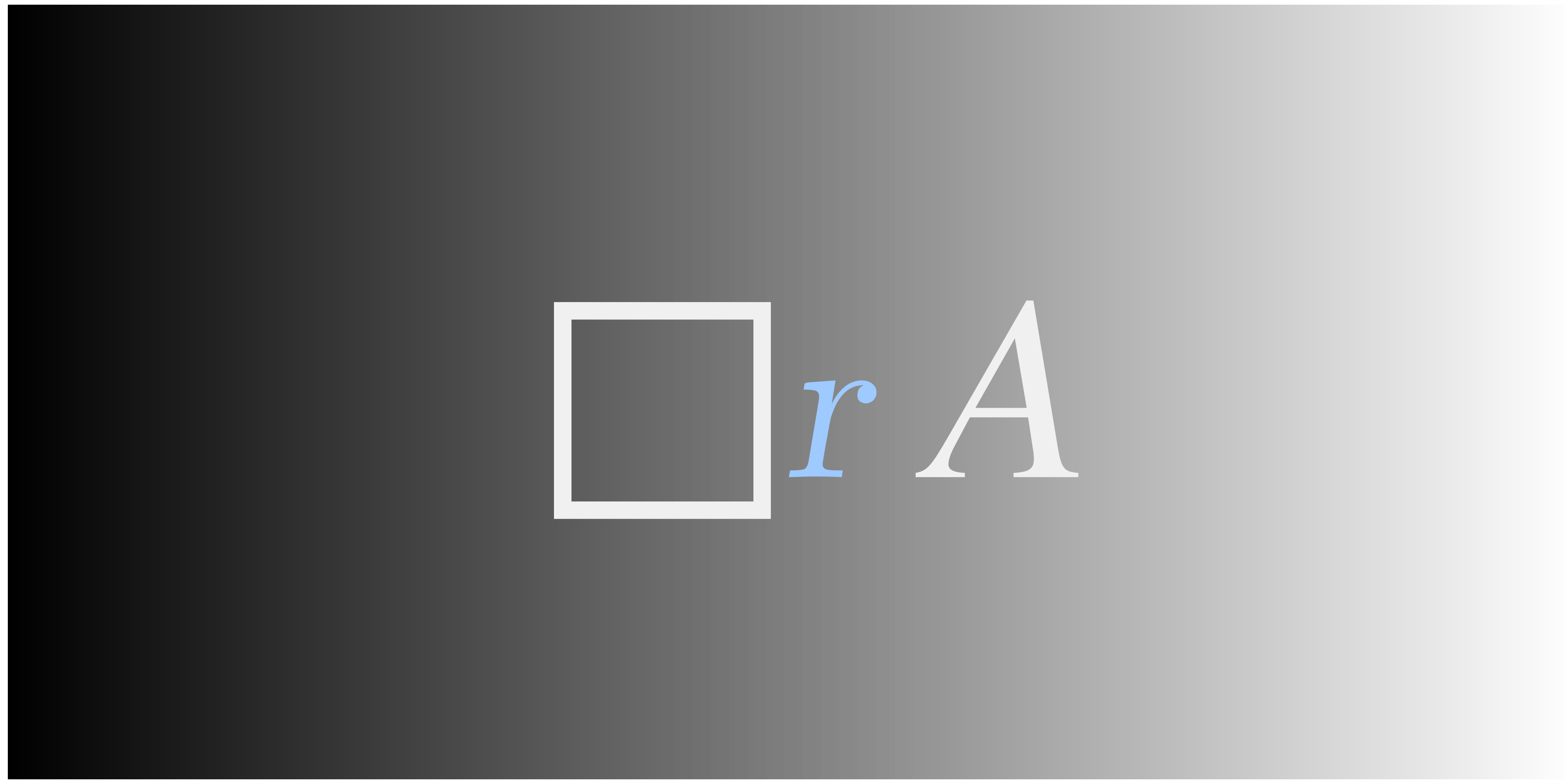
$$\frac{\Gamma_1 \vdash t_1 : A \multimap B \qquad \Gamma_2 \vdash t_2 : A}{\Gamma_1, \Gamma_2 \vdash t_1\, t_2 : B}\ \multimap_e \qquad \frac{\Gamma \vdash t : B}{\Gamma, x : [A] \vdash t : B}\ \text{weak} \qquad \frac{\Gamma_1 \vdash t_1 : \square A \qquad \Gamma_2, x : [A] \vdash t_2 : B}{\Gamma_1, \Gamma_2 \vdash \text{let } [x] = t_1 \text{ in } t_2 : B}\ \square_e$$

Modal Type Analysis



linear non-linear

Graded Modal Type Analysis



$r \in \mathcal{R}$

semiring

linear non-linear

Linear logic + !-modality

$$\Gamma ::= \varnothing \mid \Gamma, x : A \mid \Gamma, x : [A]$$
$$A ::= A \multimap A' \mid \Box A$$

$$\frac{}{x : A \vdash x : A}\ \text{ax} \qquad \frac{\Gamma_1, x : A, y : B, \Gamma_2 \vdash t : C}{\Gamma_1, y : B, x : A, \Gamma_2 \vdash t : C}\ \text{ex} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma, x : [A] \vdash t : B}\ \text{der}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x \,.\, t : A \multimap B}\ \multimap_i \qquad \frac{\Gamma, x : [A], y : [A] \vdash t : B}{\Gamma, z : [A] \vdash t[z/x][z/y] : B}\ \text{contr} \qquad \frac{[\Gamma] \vdash t : B}{[\Gamma] \vdash [t] : \Box B}\ \Box_i$$

$$\frac{\Gamma_1 \vdash t_1 : A \multimap B \qquad \Gamma_2 \vdash t_2 : A}{\Gamma_1, \Gamma_2 \vdash t_1\, t_2 : B}\ \multimap_e \qquad \frac{\Gamma \vdash t : B}{\Gamma, x : [A] \vdash t : B}\ \text{weak} \qquad \frac{\Gamma_1 \vdash t_1 : \Box A \qquad \Gamma_2, x : [A] \vdash t_2 : B}{\Gamma_1, \Gamma_2 \vdash \text{let } [x] = t_1 \text{ in } t_2 : B}\ \Box_e$$

Linear logic + $\Box_r$-modality

$r \in (\mathscr{R}, *, 1, +, 0, \sqsubseteq)$ po-semiring

$$\Gamma ::= \varnothing \mid \Gamma, x : A \mid \Gamma, x : [A]_r$$
$$A ::= A \multimap A' \mid \Box_r A$$

$$\frac{}{x : A \vdash x : A}\ \text{ax} \qquad \frac{\Gamma_1, x : A, y : B, \Gamma_2 \vdash t : C}{\Gamma_1, y : B, x : A, \Gamma_2 \vdash t : C}\ \text{ex} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma, x : [A]_1 \vdash t : B}\ \text{der}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x\,.\,t : A \multimap B}\ \multimap_i \qquad \frac{\Gamma, x : [A]_r, y : [A]_s \vdash t : B}{\Gamma, z : [A]_{r+s} \vdash t[z/x][z/y] : B}\ \text{contr} \qquad \frac{[\Gamma] \vdash t : B}{r * [\Gamma] \vdash [t] : \Box_r B}\ \Box_i$$

$$\frac{\Gamma_1 \vdash t_1 : A \multimap B \qquad \Gamma_2 \vdash t_2 : A}{\Gamma_1, \Gamma_2 \vdash t_1\, t_2 : B}\ \multimap_e \qquad \frac{\Gamma \vdash t : B}{\Gamma, x : [A]_0 \vdash t : B}\ \text{weak} \qquad \frac{\Gamma_1 \vdash t_1 : \Box_r A \qquad \Gamma_2, x : [A]_r \vdash t_2 : B}{\Gamma_1, \Gamma_2 \vdash \text{let } [x] = t_1 \text{ in } t_2 : B}\ \Box_e$$

Linear logic + $\Box_r$-modality

$r \in (\mathscr{R}, *, 1, +, 0, \sqsubseteq)$ po-semiring

$$\Gamma ::= \varnothing \mid \Gamma, x : A \mid \Gamma, x : [A]_r$$

$$A ::= A \multimap A' \mid \Box_r A$$

$$\frac{}{x : A \vdash x : A}\ \text{ax} \qquad \frac{\Gamma_1, x : A, y : B, \Gamma_2 \vdash t : C}{\Gamma_1, y : B, x : A, \Gamma_2 \vdash t : C}\ \text{ex} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma, x : [A]_1 \vdash t : B}\ \text{der}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x . t : A \multimap B}\ \multimap_i \qquad \frac{\Gamma, x : [A]_r, y : [A]_s \vdash t : B}{\Gamma, z : [A]_{r+s} \vdash t[z/x][z/y] : B}\ \text{contr} \qquad \frac{[\Gamma] \vdash t : B}{r * [\Gamma] \vdash [t] : \Box_r B}\ \Box_i$$

$$\frac{\Gamma_1 \vdash t_1 : A \multimap B \qquad \Gamma_2 \vdash t_2 : A}{\Gamma_1 + \Gamma_2 \vdash t_1\, t_2 : B}\ \multimap_e \qquad \frac{\Gamma \vdash t : B}{\Gamma, x : [A]_0 \vdash t : B}\ \text{weak} \qquad \frac{\Gamma_1 \vdash t_1 : \Box_r A \qquad \Gamma_2, x : [A]_r \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash \text{let } [x] = t_1 \text{ in } t_2 : B}\ \Box_e$$

$$(\Gamma, x : [A]_r) + (\Gamma', x : [A]_s) = (\Gamma + \Gamma'), x : [A]_{r+s}$$

Graded base and grades in the wild

Linear base

$$\Gamma ::= \emptyset \mid \Gamma, x : A \mid \Gamma, x : [A]_r$$

$$A ::= A \multimap A' \mid \Box_r A$$

Graded base

Only graded assumptions

$$\Gamma ::= \emptyset \mid \Gamma, x : [A]_r$$

$$A ::= A\, r \multimap A' \mid \Box_r A$$

Function carries a grade

Graded base rules

$$\frac{}{x : [A]_1 \vdash x : A}\ \text{ax} \qquad \frac{\Gamma, x : [A]_r \vdash t : B}{\Gamma \vdash \lambda x\,.\,t : A\, r \multimap B} \multimap_i \qquad \frac{\Gamma_1 \vdash t_1 : A\, r \multimap B \qquad \Gamma_2 \vdash t_2 : A}{\Gamma_1 + r * \Gamma_2 \vdash t_1\, t_2 : B} \multimap_e$$

- Like original coeffect type systems
- Also QTT (McBride, Atkey, Wood) and Idris 2 (Brady)

*(2014) Petricek, O, Mycroft - Coeffects: a calculus of context-dependent computation.

Linear base

$$\frac{\Gamma_1 \vdash t_1 : \Box_r A \multimap B \qquad \dfrac{[\Gamma_2] \vdash t_2 : A}{r * \Gamma_2 \vdash [t_2] : \Box_r A}\,\Box_i}{\Gamma_1 + r * \Gamma_2 \vdash t_1\ [t_2] : B}\,\Box_e$$

Graded base

$$\frac{\Gamma_1 \vdash t_1 : A\ r \multimap B \qquad \Gamma_2 \vdash t_2 : A}{\Gamma_1 + r * \Gamma_2 \vdash t_1\ t_2 : B}\,\textbf{app}$$

Same idea as before capture structure of computation/proof via grades `language GradedBase`

$A\ r \multimap B$ **written** `A % r -> B`

Graded types in Haskell (GHC 9)

```
{-# LANGUAGE LinearTypes #-}
```

cf. linear-base:

```
a %r -> b
```

Graded arrow

Linear `a %One -> b` | `a -> b`, `a [Lin] -> b`

Unrestricted `a %Many -> b` | `a [Many] -> b`

Graded modality

```
data Box r a where { Box :: a %r-> Box r a }
```

(2017) - Bernardy, Boespflug, Newton, Peyton Jones, Spiwak - Linear Haskell: practical linearity in a higher-order polymorphic language.

Graded types in Haskell one day?

```
{-# LANGUAGE GradedTypes #-}
```

```
Semiring s =>  a %(r : s) -> b
```

Challenges

- Generalise existing approach (0 missing)
- Solving inside GHC?
- Customisation? [i.e., user-defined Semiring]

Idris Graded types in Idris 2 (based on QTT)

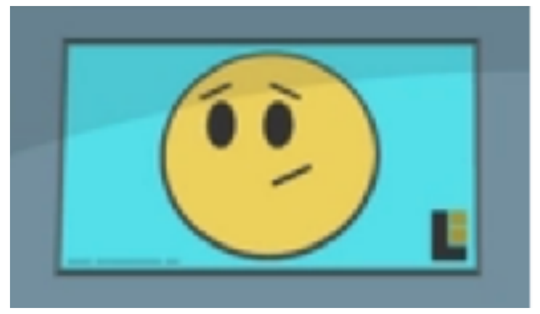
```
append : {0 n : Nat } -> {0 m : Nat } -> {0 a : Type }
       -> (1 xs : Vect n a) -> (1 ys : Vect m a) -> Vect ( n + m ) a
```

Grades drawn from $0, 1, \omega$ (the LNL semiring in Granule)

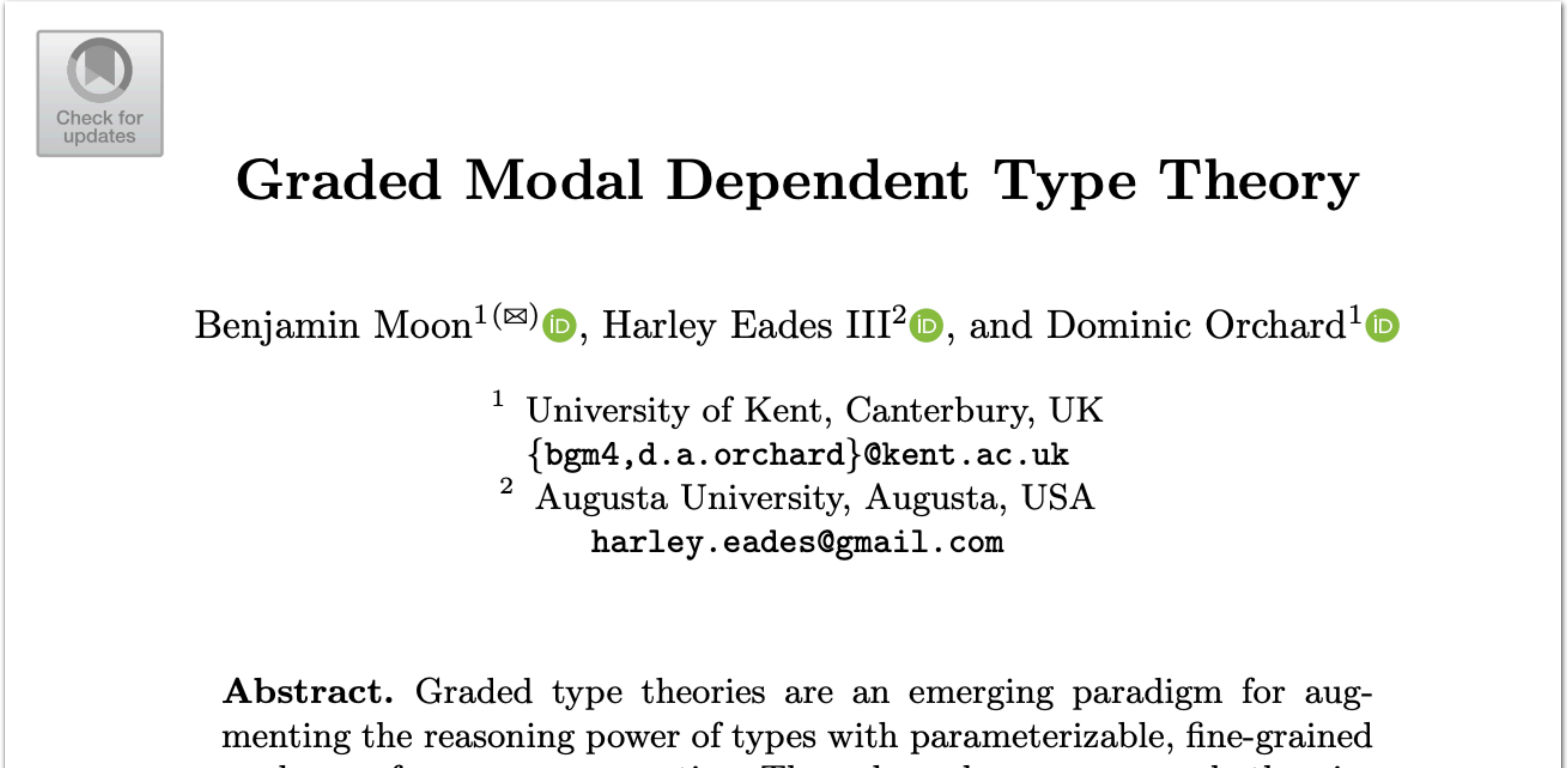
(2016) - McBride - I Got Plenty o' Nuttin'

(2018) - Atkey - Syntax and Semantics of Quantitative Type Theory

(2021) - Brady - Idris 2: Quantitative Type Theory in Practice.

Gerty  generalises this to track type- + computation- use

```
append : {n : (0,2) Nat } -> {m : (0,2) Nat } -> {a : (0,2(n+m)) Type }
       -> (xs : (1,0) Vect n a) -> (ys : (1,0) Vect m a) -> Vect (n + m) a
```



Check for updates

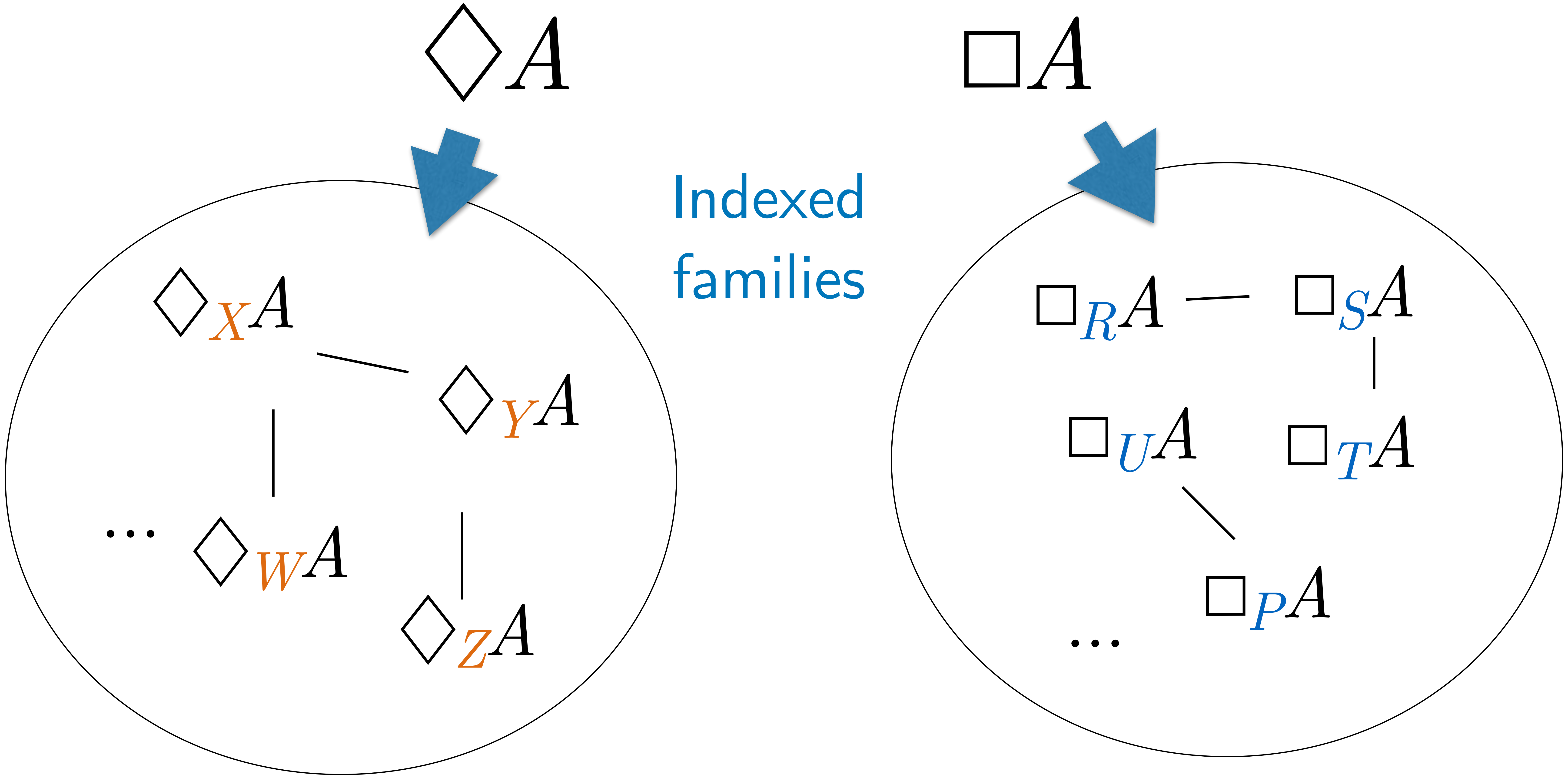
Graded Modal Dependent Type Theory

Benjamin Moon[1](✉), Harley Eades III[2], and Dominic Orchard[1]

[1] University of Kent, Canterbury, UK
{bgm4,d.a.orchard}@kent.ac.uk
[2] Augusta University, Augusta, USA
harley.eades@gmail.com

Abstract. Graded type theories are an emerging paradigm for augmenting the reasoning power of types with parameterizable, fine-grained

Graded modalities (informally)



with structure
matching the shape of proofs/programs or a semantics

Possibility / monads

Hilbert-style

$$\Diamond (A \to B) \to \Diamond A \to \Diamond B$$ *(K) distributivity*

$$A \to \Diamond A$$ *(T) reflexivity*

$$\Diamond\Diamond A \to \Diamond A$$ *(4) transitivity*

Natural deduction (+ terms):

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash \textbf{return}\ e : \Diamond A} \qquad \frac{\Gamma \vdash e_1 : \Diamond A \qquad \Gamma, x : A \vdash e_2 : \Diamond B}{\Gamma \vdash \textbf{do}\ x \leftarrow e_1; e_2 : \Diamond B}$$

Benton, Bierman, de Paiva - Computational Types from a Logical Perspective - JFP (1998)

Graded possibility / monads

$x \in (X, \circledast, I)$ is a monoid

Hilbert-style

$$\Diamond_x (A \to B) \to \Diamond_x A \to \Diamond_x B \qquad \textit{(K) distributivity}$$

$$A \to \Diamond_I A \qquad \textit{(T) reflexivity}$$

$$\Diamond_x \Diamond_y A \to \Diamond_{x \circledast y} A \qquad \textit{(4) transitivity}$$

Natural deduction (+ terms):

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash \textbf{return } e : \Diamond_I A} \qquad \frac{\Gamma \vdash e_1 : \Diamond_x A \qquad \Gamma, x : A \vdash e_2 : \Diamond_y B}{\Gamma \vdash \textbf{do } x \leftarrow e_1; e_2 : \Diamond_{x \circledast y} B}$$

Katsumata - Parametric effect monads and semantics of effect systems (2014)

O, Petricek, Mycroft - The semantic marriage of effects and monads (2014)

$\Diamond x\ A$ written in Granule as `A < x >`

Effect-set-graded possibility

$$(X, \circledast, I) = (\mathscr{P}(\text{IOlabels}), \cup, \emptyset)$$

$\mathbb{N}$-graded possibility

$$(X, \circledast, I) = (\mathbb{N}, +, 0)$$

Graded monads

in programming

https://hackage.haskell.org/package/effect-monad

```
class GMonad (g :: k -> Type -> Type) where
  return :: a -> g Zero a
  (>>=)  :: g x -> (a -> g y b) -> g (Plus x y) b
```

```
put :: Var v -> s -> State {v :-> W ! s} ()
get :: Var v -> State {v :-> R ! s} s
```

in semantics

$$\Gamma \vdash e : A?F \xrightarrow{[\![-]\!]} ([\![\Gamma]\!] \to M_F[\![A]\!])$$

Type-and-effect systems

Orchard and Petricek - Embedding effect systems in Haskell (2014)

Trivial vs "meaningful" graded monads

Purely for analysis / controlling expressivity

$$[\![\Diamond_x A]\!] = [\![A]\!]$$

For analysis but semantics unrefined

$$[\![\Diamond_x A]\!] = M\,[\![A]\!]$$

e.g. $[\![\Diamond_x A]\!] = S \rightarrow [\![A]\!] \times S$

Graded-directed semantics

e.g. $[\![\Diamond_x A]\!] = \mathsf{reads}(x) \rightarrow [\![A]\!] \times \mathsf{writes}(x)$

e.g. $x = \{\mathit{read}(a), \mathit{read}(b), \mathit{write}(b)\}$

Liveness graded state monad

$$\mathsf{Gm}^{\psi} A = \mathsf{Store}(\mathbf{liveIn}(\psi)) \to A \times \mathsf{Store}(\mathbf{footprint}(\psi))$$

FSCD 2020

Data-Flow Analyses as Effects and Graded Monads

Andrej Ivašković
Department of Computer Science and Technology, University of Cambridge, UK
andrej.ivaskovic@cst.cam.ac.uk

Alan Mycroft
Department of Computer Science and Technology, University of Cambridge, UK
alan.mycroft@cst.cam.ac.uk

Dominic Orchard
School of Computing, University of Kent, UK
d.a.orchard@kent.ac.uk

Abstract

In static analysis, two frameworks have been studied extensively: monotone data-flow analysis and type-and-effect systems. Whilst both are seen as general analysis frameworks, their relationship has remained unclear. Here we show that monotone data-flow analyses can be encoded as effect systems in a uniform way, via algebras of transfer functions. This helps to answer questions about the

Semiring-graded necessity:

$$\Box_{r*s} A \to \Box_r \Box_s A$$

$$\Box_1 A \to A$$

$$\Box_r (A \to B) \to \Box_r A \to \Box_r B$$

$$\Box_0 A \to 1$$

$$\Box_{r+s} A \to \Box_r A \otimes \Box_s A$$

} plumbing dataflow

Monoid-graded possibility:

$$\Diamond_x \Diamond_y A \to \Diamond_{x \circledast y} A$$

$$A \to \Diamond_I A$$

$$\Diamond_x (A \to B) \to \Diamond_x A \to \Diamond_x B$$

Coeffects

Effects

Asymmetry is because λ-calculus input has more structure

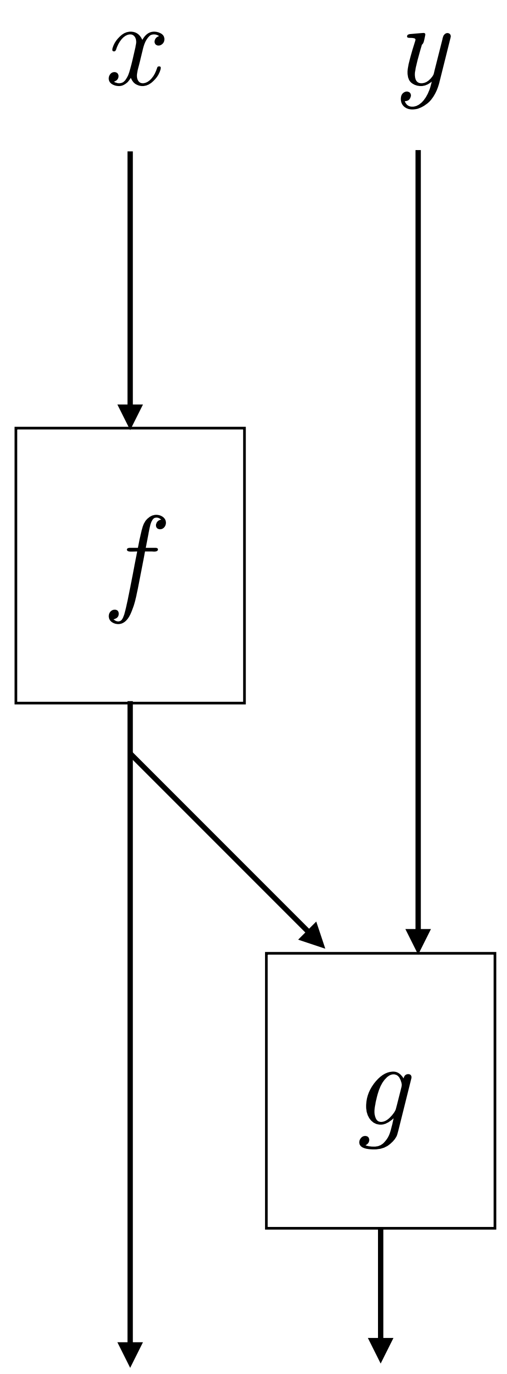
$$\Gamma \vdash e : \tau$$

many to one

Dataflow analysis via grading

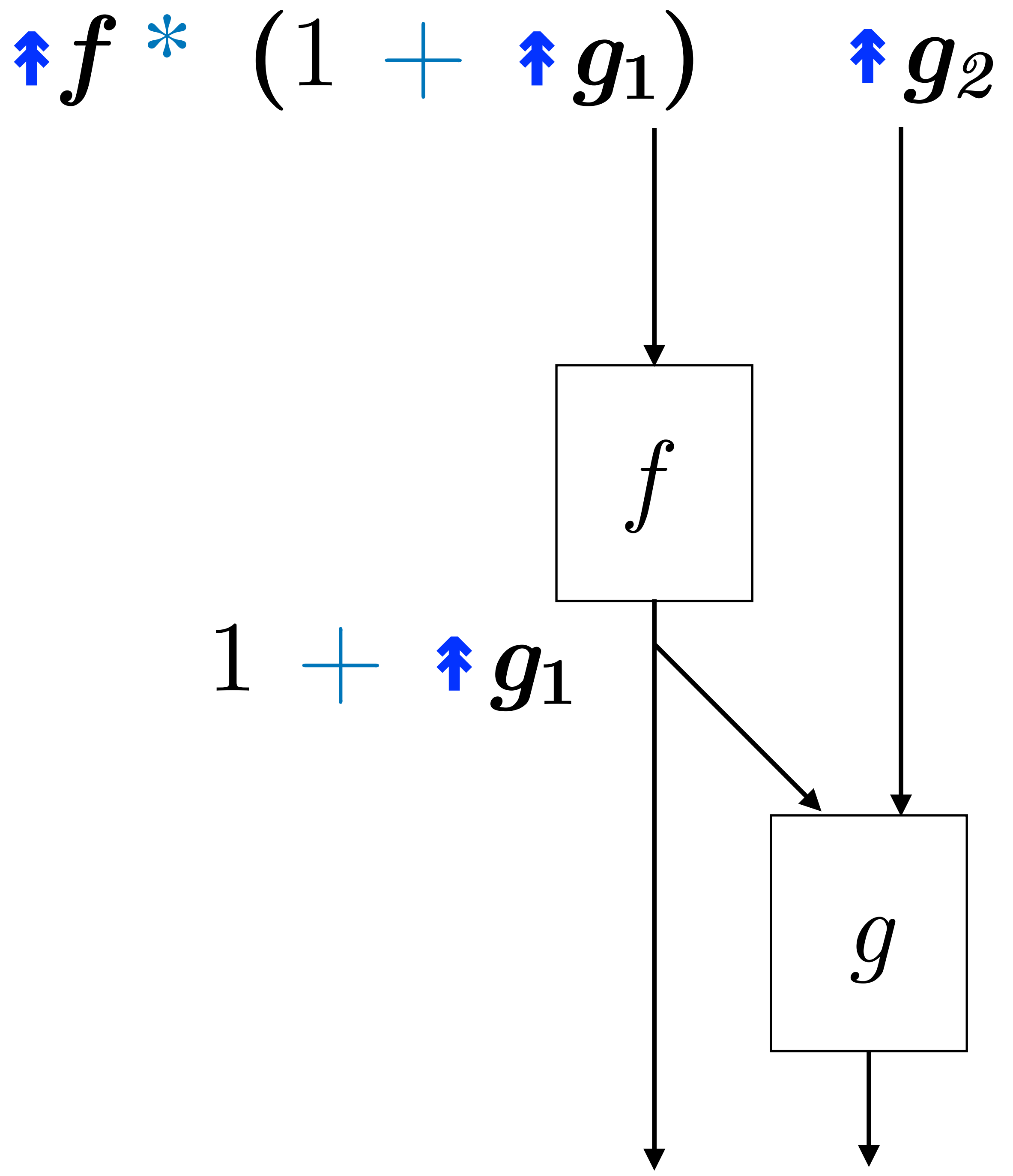
Consider $\lambda x\ .\ \lambda y\ .\ \textbf{let}\ z = f\ x\ \textbf{in}\ (z,\ g\ z\ y)$

Flow graph



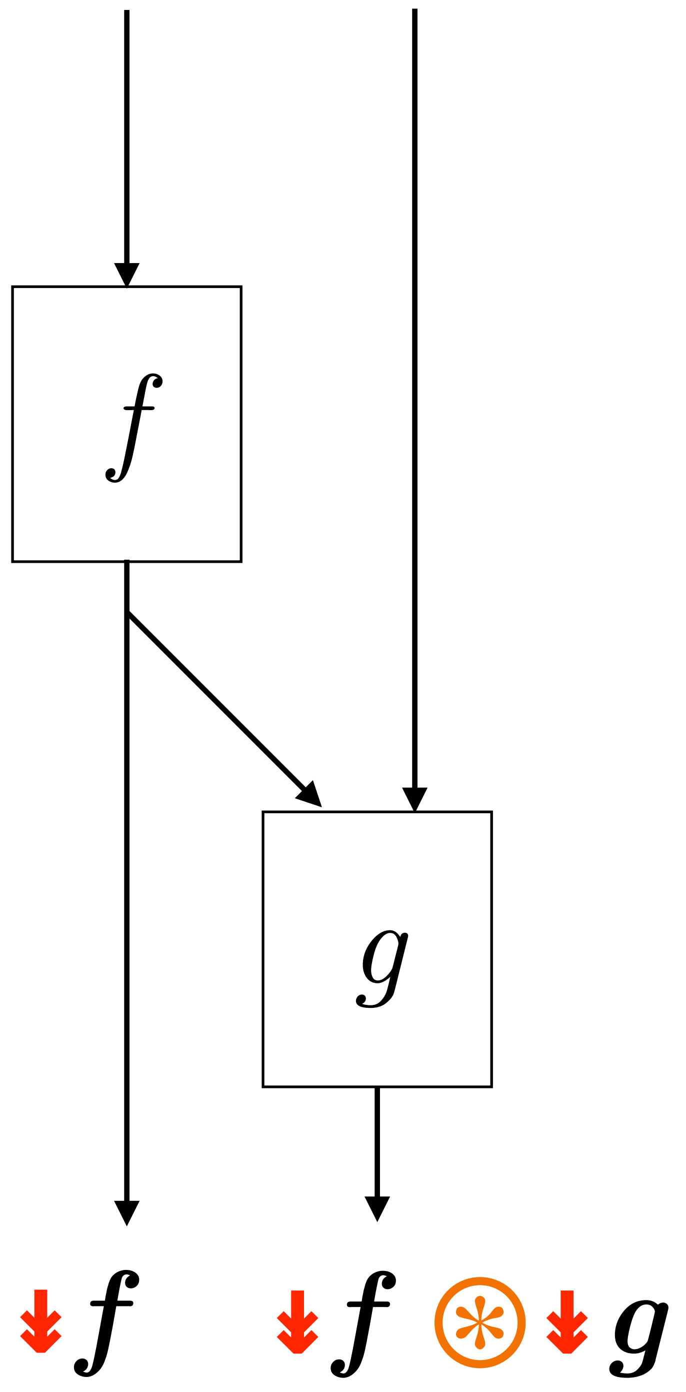
Backward

$\twoheaduparrow \boldsymbol{h_i} = \textit{demands on } i^{th} \textit{ parameter to } \boldsymbol{h}$



Forward

$\twoheaddownarrow \boldsymbol{h} = \textit{provision of } \boldsymbol{h}$



Other instances of graded modalities

- Contextual Model Type Theory - CMTT (Nanevski et al. ‘08)

  $\Box_{\Gamma} A$ meaning A is true under closure of Γ

- Hardware schedules (Ghica et al. ’14)
- Explicit provability logics (Artemov ’95, ’01)
- Multi-stage programming (generalising Pfenning & Davies, ’01)
- Costs (cf. Cicek et al. 17)
- Robustness / sensitivity (Gaboardi et al. ‘16, Pierce et al. ’13)
- Provenance
- Probabilistic programming (forwards / backwards)
- Type state (stateful protocols)

What did we learn?

- Defining typing theories declaratively / formally
- We implicitly heavily leveraged Curry-Howard
- Linear types for resourceful thinking
- Modal reasoning
- Graded modal reasoning (in three flavours)
- Now what?
    - Do you have a binary property that you can make more fine grained?

Graded slogan: match the structure of proof/program with structure in the indices

Thanks!