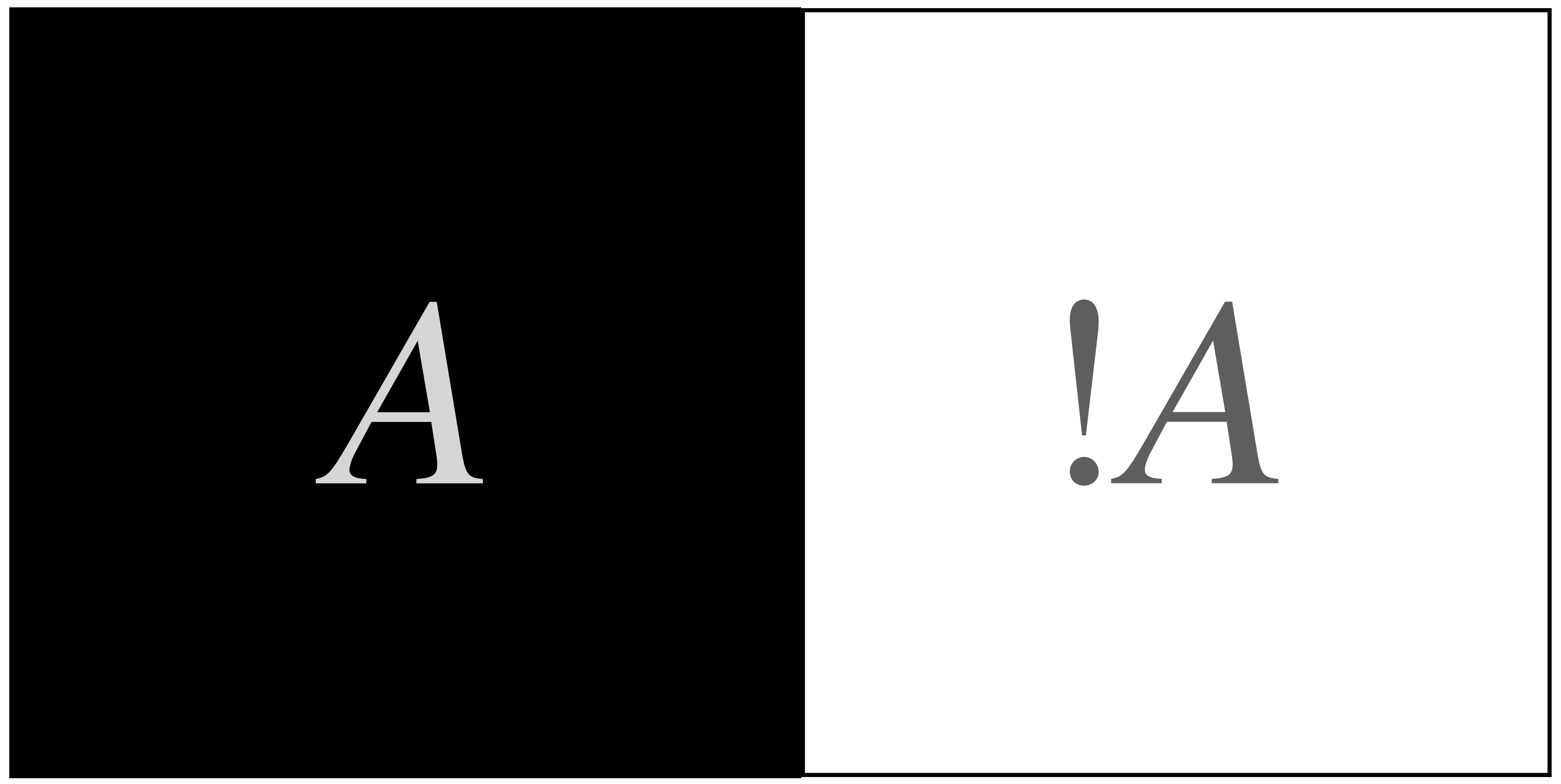
Graded Types - Part 2

Extending from linear types to graded types

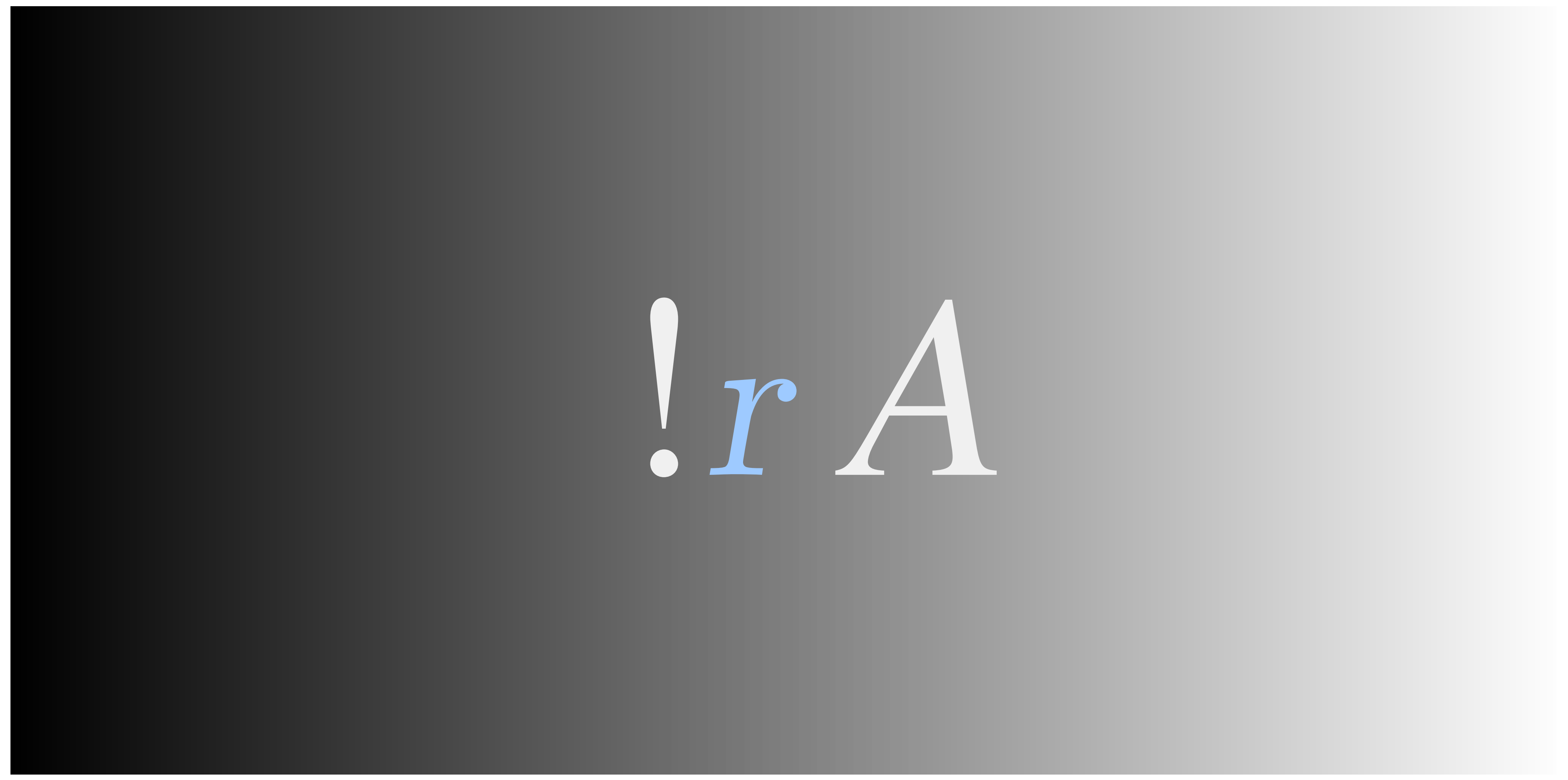
Dominic Orchard, 24-28th July 2023, SPLV23

Modal Type Analysis



linear non-linear

Graded Modal Type Analysis



$r \in \mathscr{R}$

semiring

linear non-linear

! modality — use any number of times

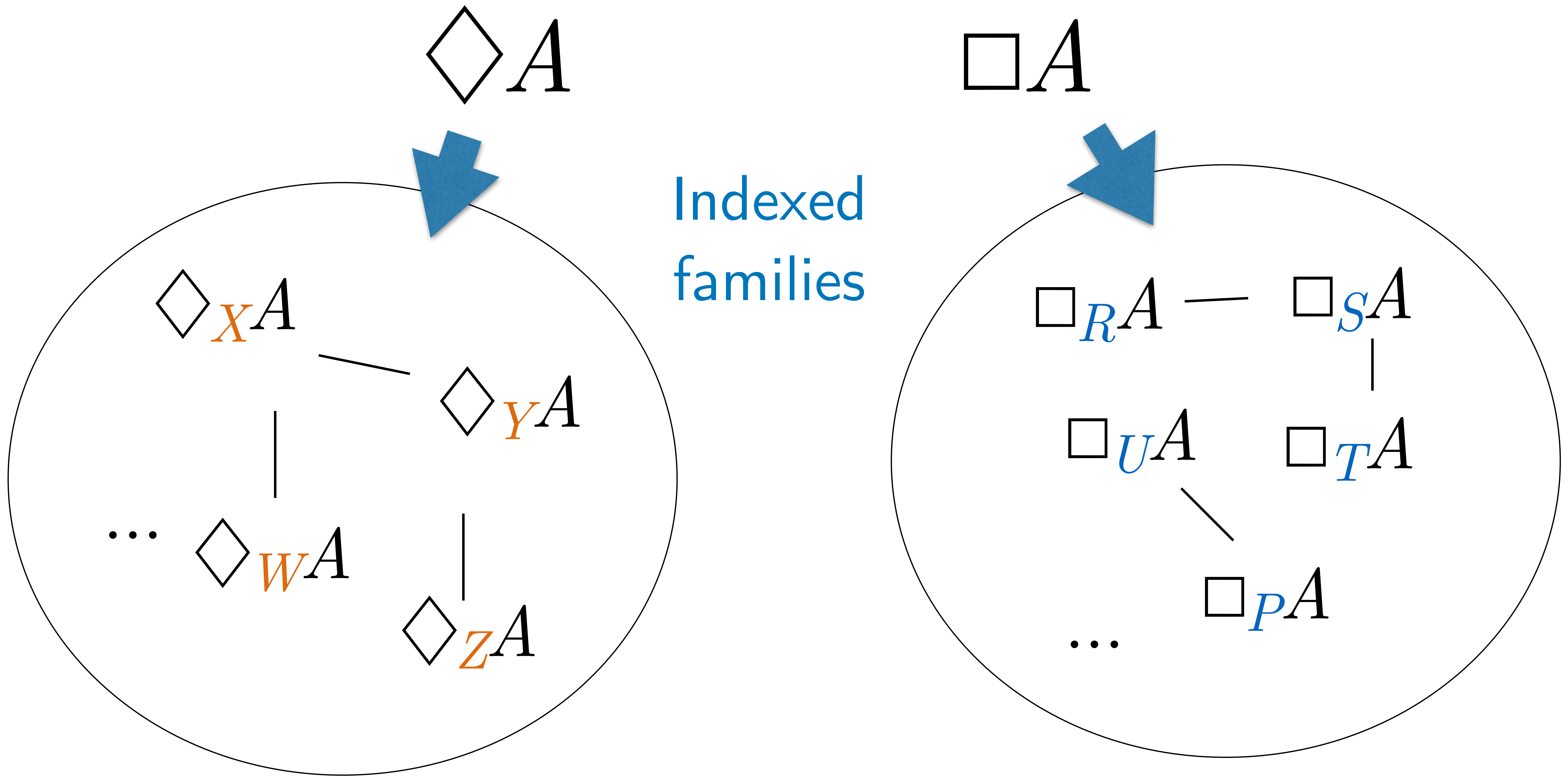
linear logic — use exactly once

□ω modality — use any number of times

□n modality — use at most n number of times

linear types — use exactly once

Graded modalities (informally)



with structure

matching the shape of proofs/programs or a semantics

Graded modal types (graded necessity) in Granule

Quantitative Program Reasoning with Graded Modal Types

DOMINIC ORCHARD, University of Kent, UK
VILEM-BENJAMIN LIEPELT, University of Kent, UK
HARLEY EADES III, Augusta University, USA

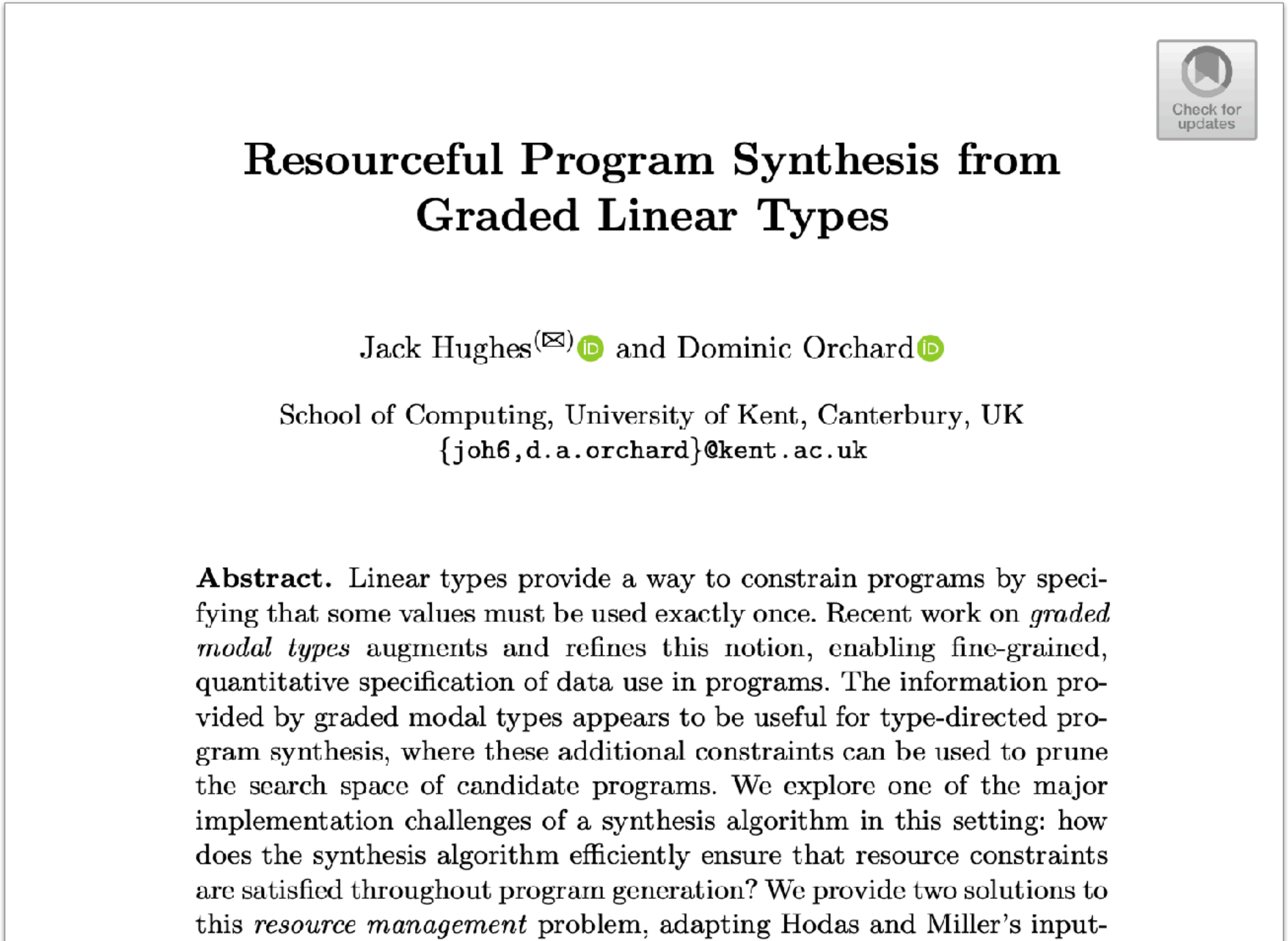
In programming, some data acts as a resource (e.g., file handles, channels) subject to usage constraints. This poses a challenge to software correctness as most languages are agnostic to constraints on data. The approach of linear types provides a partial remedy, delineating data into resources to be used but never copied or discarded, and unconstrained values. Bounded Linear Logic provides a more fine-grained approach, quantifying non-linear use via an indexed-family of modalities. Recent work on *coeffect types* generalises this idea to *graded comonads*, providing type systems which can capture various program properties. Here, we propose the umbrella notion of *graded modal types*, encompassing coeffect types and dual notions of type-based effect reasoning via *graded monads*. In combination with linear and indexed types, we show that graded modal types provide an expressive type theory for quantitative program reasoning, advancing the reach of type systems to capture and verify a broader set of program properties. We demonstrate this approach via a type system embodied in a fully-fledged functional language called Granule, exploring various examples.

CCS Concepts: • **Theory of computation** → **Modal and temporal logics**; **Program specifications**; **Program verification**; *Linear logic*; *Type theory*.

Additional Key Words and Phrases: graded modal types, linear types, coeffects, implementation

ACM Reference Format:
Dominic Orchard, Vilem-Benjamin Liepelt, and Harley Eades III. 2019. Quantitative Program Reasoning

LOPSTR 2020



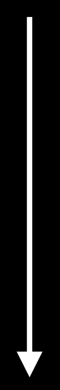
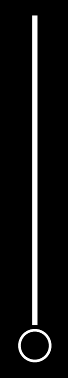
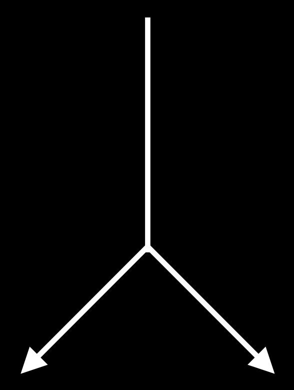
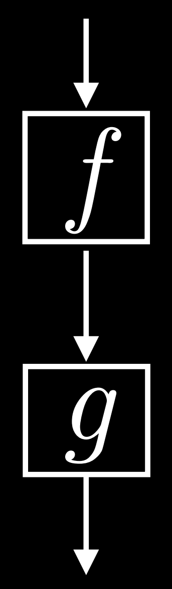
Resourceful Program Synthesis from Graded Linear Types

Jack Hughes(✉) and Dominic Orchard

School of Computing, University of Kent, Canterbury, UK
{joh6,d.a.orchard}@kent.ac.uk

Abstract. Linear types provide a way to constrain programs by specifying that some values must be used exactly once. Recent work on *graded modal types* augments and refines this notion, enabling fine-grained, quantitative specification of data use in programs. The information provided by graded modal types appears to be useful for type-directed program synthesis, where these additional constraints can be used to prune the search space of candidate programs. We explore one of the major implementation challenges of a synthesis algorithm in this setting: how does the synthesis algorithm efficiently ensure that resource constraints are satisfied throughout program generation? We provide two solutions to this *resource management* problem, adapting Hodas and Miller's input-

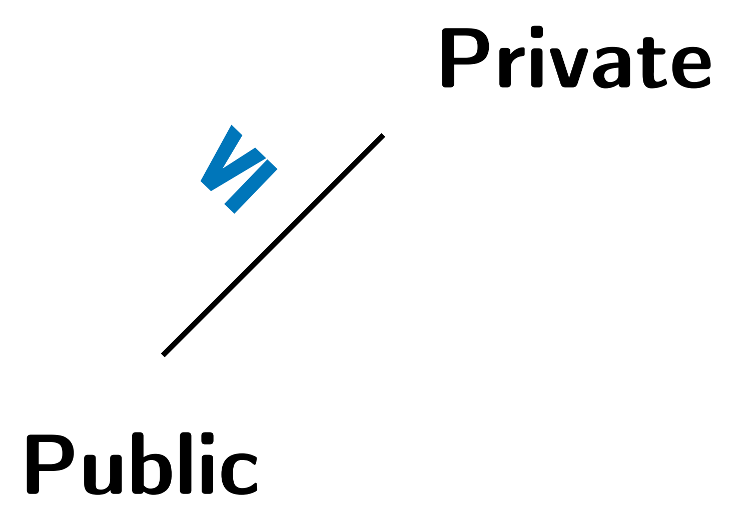
Semiring graded modalities capture dataflow

Linear	Weaken	Contract	Compose
			
use	discard	split	sequence
1	**0**	**+**	*****

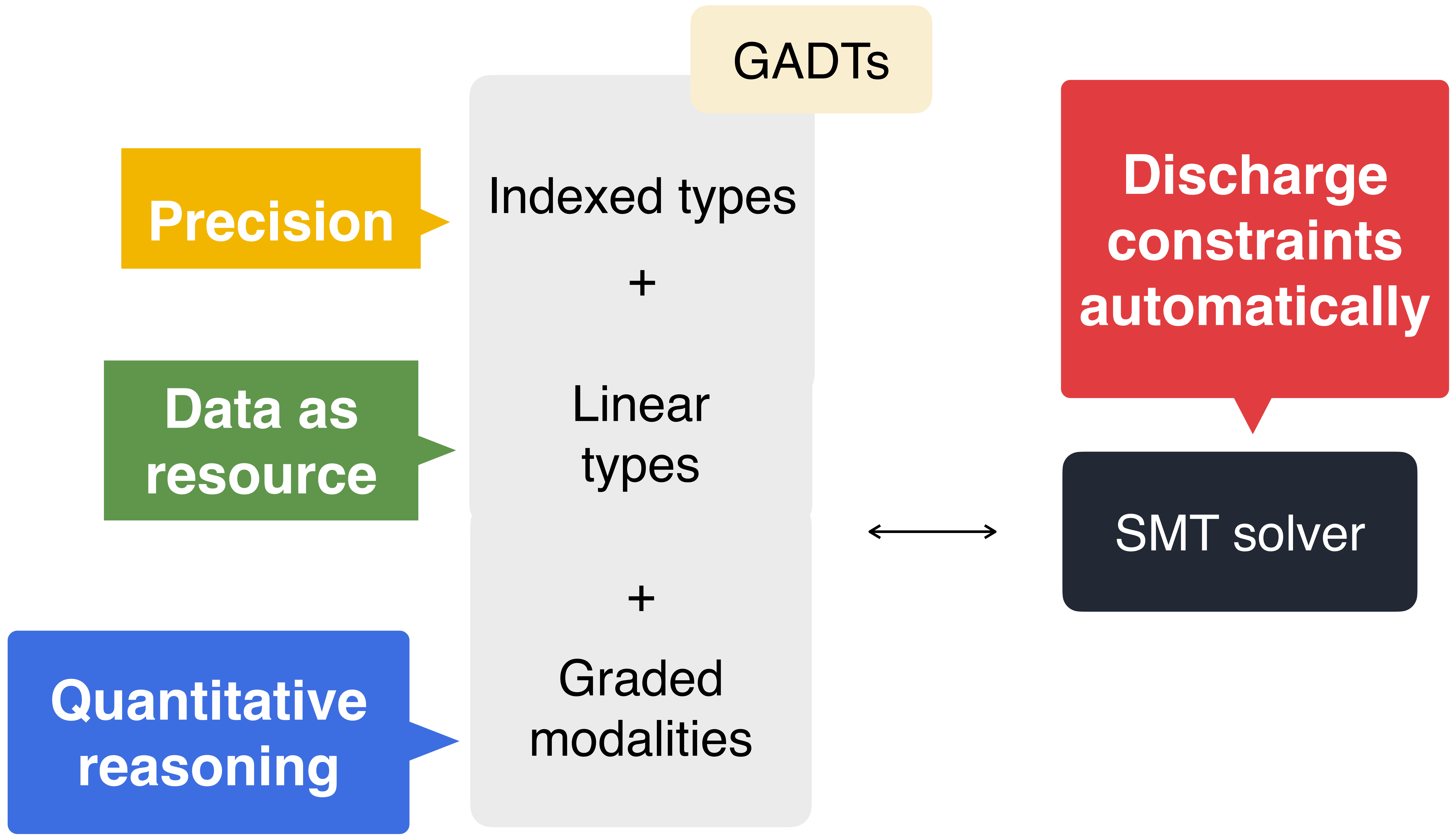
Grading algebra (semiring) captures dataflow

(In case I forget): Next demo

Security levels



The granule language



Linear types + graded modality

$r \in (\mathscr{R}, *, 1, +, 0)$ **is a semiring**

$$A, B ::= A \to B \mid \Box_r A$$

Non-linear value of type A

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

Non-linear variable x of type A

e.g.

$$\frac{x : [A]_2 \vdash (x, x) : A \otimes A}{\varnothing \vdash \lambda[x] \,.\, (x, x) : \Box_2 A \to A \otimes A}$$

(2013) Petricek, O, Mycroft - Coeffects: Unified Static Analysis of Context-Dependence

(2014) Ghica, Smith - Bounded linear types in a resource semiring

(2014) Brunel, Gaboardi, Mazza, Zdancewic - A Core Quantitative Coeffect Calculus

Linear types + graded modality

$\boldsymbol{r} \in (\mathscr{R}, *, \mathbf{1}, +, \mathbf{0})$ **is a semiring**

$$\frac{\Gamma \vdash t : B}{\Gamma, x : [A]_0 \vdash t : B}\ \text{weak} \qquad \frac{\Gamma_1 \vdash t : A \multimap B \quad \Gamma_2 \vdash t' : A}{\Gamma_1 + \Gamma_2 \vdash t\, t' : B}\ \text{app}$$

Use anytime we need to combine contexts

contraction

$$\begin{cases} \Gamma_1 + (\Gamma_2, x : A) = (\Gamma_1 + \Gamma_2), x : A & \textbf{if } x \notin |\Gamma_1| \\ \Gamma_1, x : A + \Gamma_2 = (\Gamma_1 + \Gamma_2), x : A & \textbf{if } x \notin |\Gamma_2| \end{cases}$$

$$(\Gamma_1, x : [A]_r) + (\Gamma_2, x : [A]_s) = (\Gamma_1 + \Gamma_2), x : [A]_{r+s}$$

Modal rule 1 - Dereliction

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma, x : [A]_1 \vdash t : B}\ \text{der}$$

Treat a linear variable as non-linear (dereliction)

Modal rule 2 - Promotion

$$\frac{[\Gamma] \vdash t : B}{r * [\Gamma] \vdash [t] : \Box_r B}\ \text{pr}$$

Non-linear results require non-linear variables (promotion)

Modal rule 3 - Cut

$$\frac{\Gamma \vdash t_1 : \Box_r A \qquad \Delta, x : [A]_r \vdash t_2 : B}{\Gamma + \Delta \vdash \mathsf{let}\, [x] = t_1 \,\mathsf{in}\, t_2 : B}\ \text{cut}$$

Composition (substitution) of non-linear value into non-linear variable

A core quantitative coeffect calculus [Brunel et al.'14]

Nested pattern matching and grades

```
push : forall {a b : Type, n : Nat}
     . (a, b) [n] -> (a [n], b [n])

push [(x, y)] = ([x], [y])
```

Binds $x : [A]_n, y : [B]_n$ (which Granule reports using notation: `x  :  .[a].  [n]`...)

Inner patterns inherit grade of outer patterns

Nested pattern matching and grades

```
push : forall {a b : Type, n : Nat}
     . (a, b) [n] -> (a [n], b [n])

push [(x, y)] = ([x], [y])
```

But … linear logic does not permit $!(A \otimes B) \multimap\ !A \otimes !B$

Instead… partial operation added to act as predicate

```
push : forall {a b : Type, s : Semiring, r : s}
     . {r ≚ r} => (a, b) [r] -> (a [r], b [r])

push [(x, y)] = ([x], [y])
```

(2021) Hughes et al.: Linear Exponentials as Graded Modal Types

Two layers of grading

```
f : (Vec … Patient) [0..1] -> …

f [Cons (Patient [city] [_]) ] = …
```

Public   0..1

Generates the context

```
city : .[String]. ([0..1] × Public)
```

Semirings

```
Nat       : Semiring
Level     : Semiring                  {Private, Public} or {Hi, Lo}
Q         : Semiring
LNL       : Semiring                  {Zero, One, Many}
Cartesian : Semiring                  {Any}
```

(see examples/scale.gr)

```
Set       : Type -> Semiring
SetOp     : Type -> Semiring
Ext       : Semiring -> Semiring
Interval  : Semiring -> Semiring
_×_       : Semiring -> Semiring -> Semiring
```

(see examples/sets.gr)

(Ext $\mathscr{R}$ = $\mathscr{R} \cup \{\infty\}$)

Recovering $!A$ as a graded monad

Semiring $\mathscr{R} = \{\text{Zero}, \text{One}, \text{Many}\}$

$$!A = \square_{\text{Many}}\, A$$

Semiring-graded necessity captures graded comonads

Axioms:

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

$\Box_0 A \rightarrow 1$

$\Box_1 A \rightarrow A$

$\Box_{r+s} A \rightarrow \Box_r A \wedge \Box_s A$

$\Box_r (A \rightarrow B) \rightarrow \Box_r A \rightarrow \Box_r B$

$\Box_s A \rightarrow \Box_r A \quad \textit{where } r \leq s$

Model: **exponential graded comonad**

All of these are derivable from the rules

(2013) Petricek, O, Mycroft - Coeffects: Unified Static Analysis of Context-Dependence

(2014) Ghica, Smith - Bounded linear types in a resource semiring

(2014) Brunel, Gaboardi, Mazza, Zdancewic - A Core Quantitative Coeffect Calculus

Session 1 - 2

Learning plan

Learn about linear types

Learn how a type system is formally specified

Specifically: linear types for the *lambda calculus*

See examples of linear programs in Granule

Learn about (a particular flavour of) modalities and graded modalities