Resource-Oriented Programming with Graded Modal Types

Dominic Orchard Vilem-Benjamin Liepelt Harley Eades III

University of Kent

AUGUSTA UNIVERSITY

https://granule-project.github.io/

Data

~~Infinitely copiable~~

~~Arbitrarily discardable~~

~~Universally unconstrained~~

Problem

(Some) data acts as a resource

Ignoring this leads to bugs!

Solution

Capture resource constraints in types

Do this in an extensible way

granule's type system

Precision

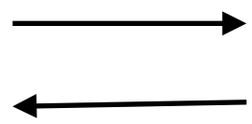
Track information for quantitative reasoning

Be strict about resources

Indexed types

Graded modal types

Linear types



SMT solver (e.g. Z3)

Why linearity?

- Efficiency
  - In-place update
- Usage protocols & resources
  - File handles*
  - Streaming I/O
  - Session typed channels*
  - Language interoperability
  - Low-level stateful protocols
  - Pointers/References**
  - Mutable arrays**
  - Circuit modeling
- Program reasoning

*implemented in Granule

**experimental support / work in progress

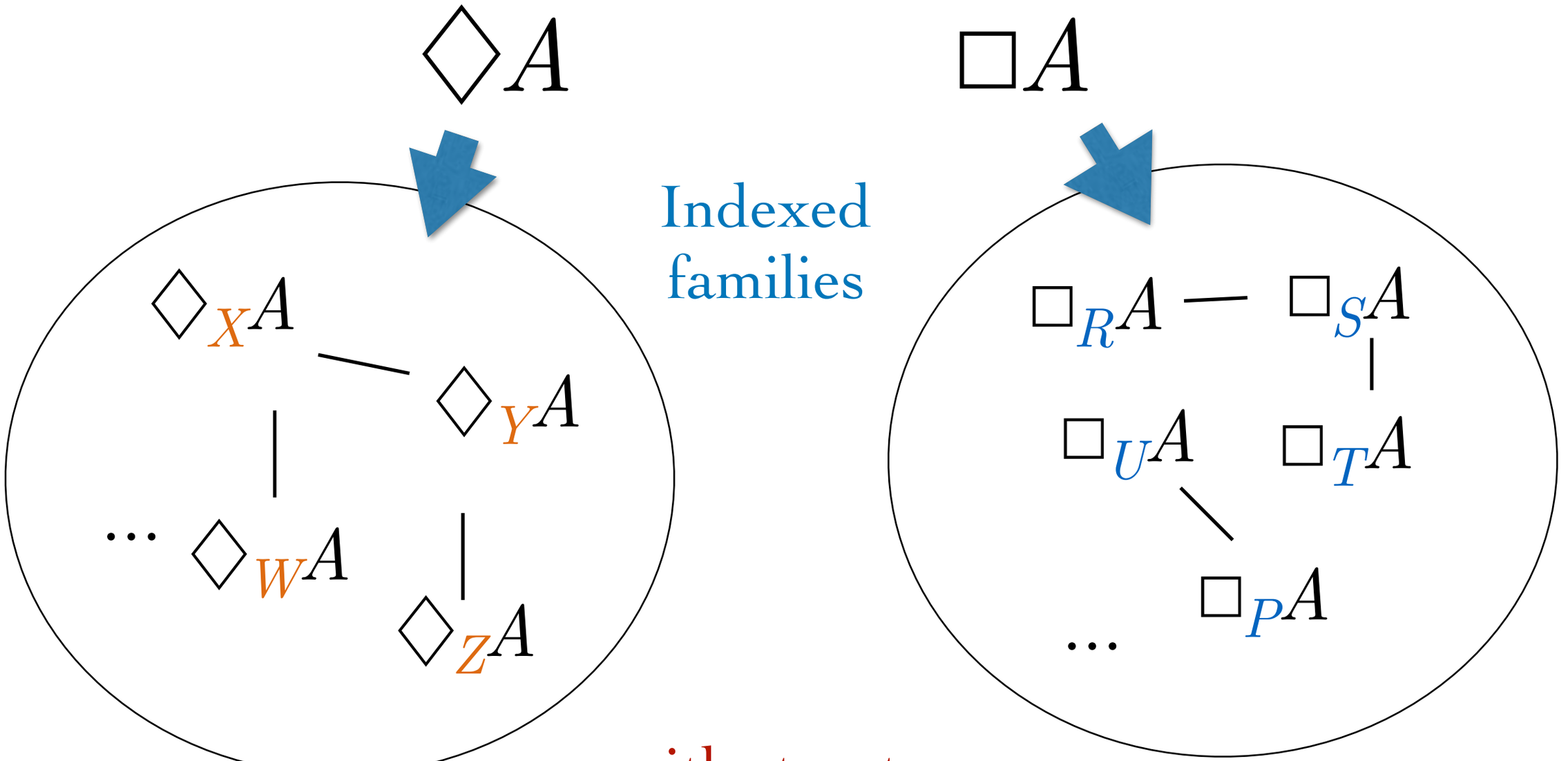
Initial Demo

Linearity, modalities, and two graded modalities

$\Box_n A$ **where** $n \in \mathbb{N}$

$\Box_{[n..m]} A$ **where** $n, m \in \mathbb{N} \wedge n \leq m$

Graded modalities (informally)



with structure

matching the shape of proofs/programs or a semantics

Graded necessity (coeffects/graded comonads)

$A\,[0] \longrightarrow ()$ weakening

$A\,[m + n] \longrightarrow (A\,[m] \times A\,[n])$ contraction

$A\,[1] \longrightarrow A$ use (dereliction)

$A\,[m * n] \longrightarrow (A\,[m])\,[n]$ composition (promotion)

$A\,[m] \longrightarrow A\,[n] \quad (\text{where } m \geq n)$ approximation

Graded by pre-ordered semiring

Graded necessity in Granule

- (Extended) naturals
- Intervals
- Security levels
- Products
- (Extended) positive rationals
- (In progress) Monotonicity (à la Arntzenius & Krishnaswami)
- <insert your idea here>

Dataflow paths

Non linear

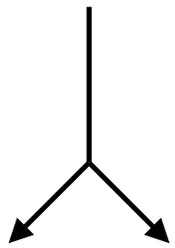
Linear

use

Weakening

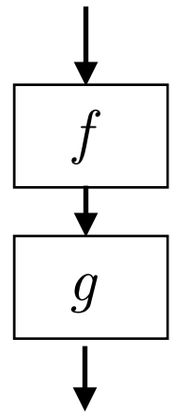
discard

Contraction



split

Composition



1 **0** **+** ***

Grading semiring captures (backwards) dataflow

THE GREAT THEOREM PROVER SHOWDOWN

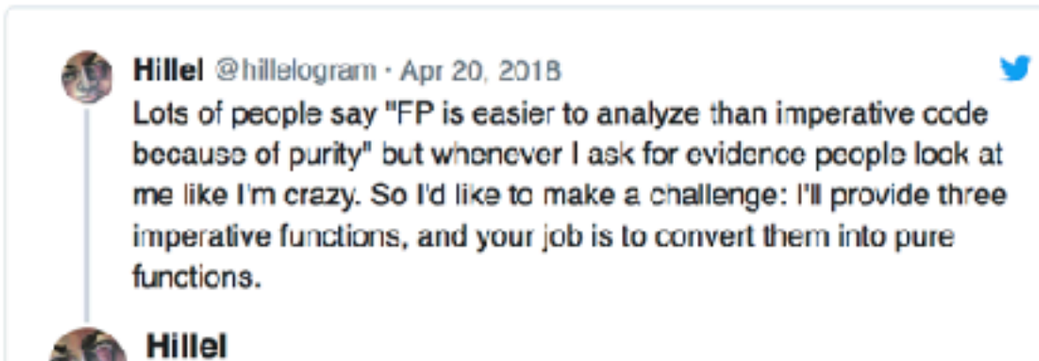
Apr 25, 2018

Functional programming and immutability are hot right now. On one hand, this is pretty great as there's lots of nice things about functional programming. On the other hand,

The Questions

> 1. *Leftpad. Takes a padding character, a string, and a total length, returns the string padded to that length with that character. If length is less than the length of the string, does nothing.*

I don't like accepting things as axioms. If we make a claim, we better damn well put it to the test. So last Friday I did exactly that:

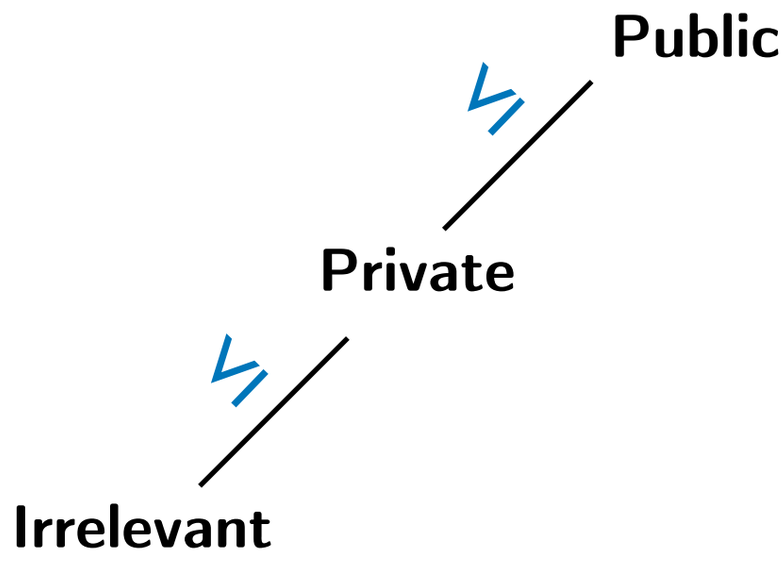


Next demo

Verification via first-class grades and indexed types

Next demo

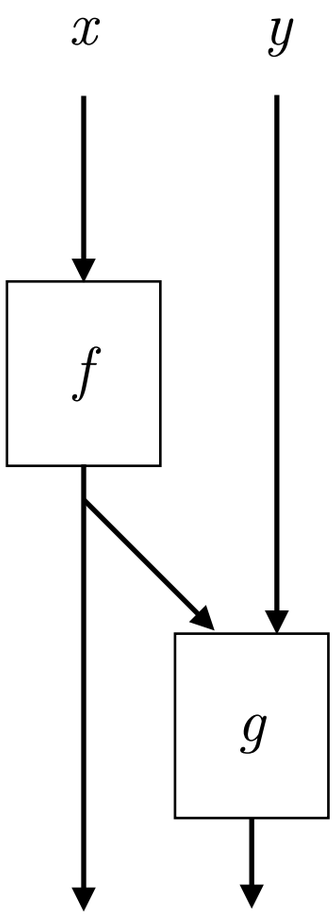
Security levels



Dataflow analysis via grading

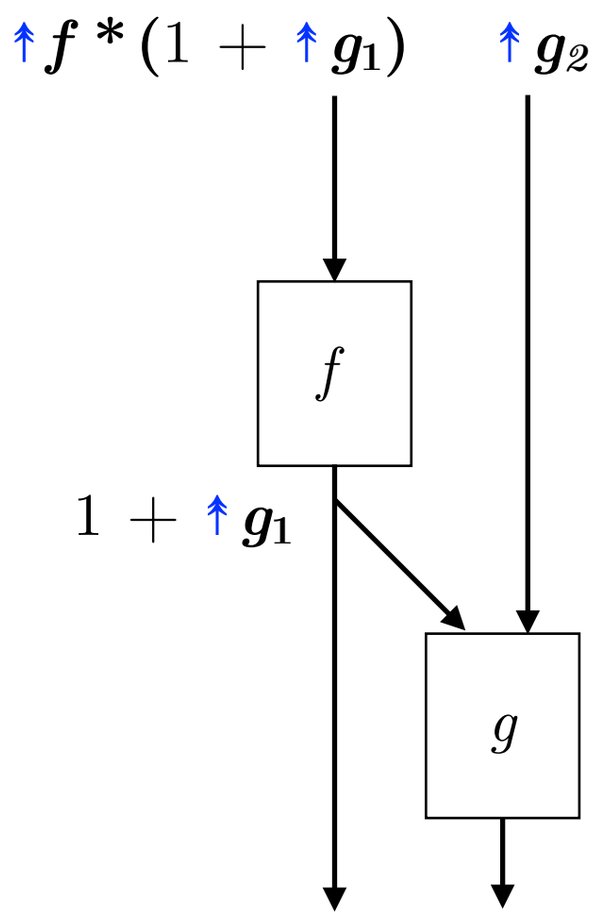
Consider $\lambda x\ .\ \lambda y\ .\ \textbf{let}\ z = f\ x\ \textbf{in}\ (z,\ g\ z\ y)$

Flow graph



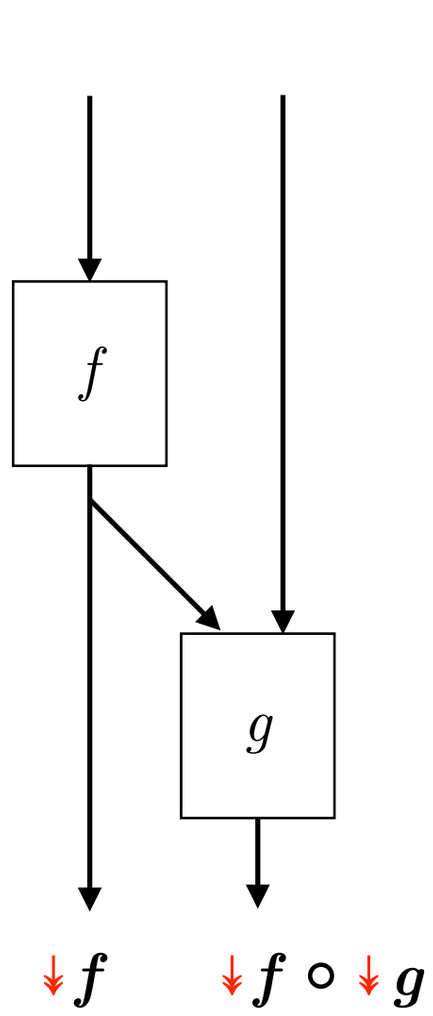
Backward

$\uparrow \boldsymbol{h_i}$ = *demands on* i^{th} *parameter to* $\boldsymbol{h}$



Forward

$\downarrow \boldsymbol{h}$ = *provision of* $\boldsymbol{h}$



Last demo

(monoid) graded possibility

Potted history of graded types

Key

graded monads
graded comonads
both

2008

Durov - New approach to Arkelov geometry

Smirnov - Graded monads and rings of polynomials

2013

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

2014

Ghica, Smith - Bounded linear types in a resource semiring

Brunel, Gaboardi, Mazza, Zdancewic - A Core Quantitative Coeffect Calculus

Katsumata - Parametric effect monads and semantics of effect systems.

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

Petricek, O, Mycroft - Coeffects: a calculus of context-dependent computation.

2016

Gaboardi, Katsumata, O, Breuvart, Uustalu - Combining effects & coeffects via grading

.... various coeffect papers

2019

O, Liepelt, Eades - Quantitative program reasoning with graded modal types

Come see Granule at ICFP 2019

https://granule-project.github.io/