# Quantitative program reasoning in Granule via graded modal types

CODE MESH LDN

Dominic Orchard

University of Kent

# The Granule Project

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


University of Kent

Dominic Orchard

Vilem Liepelt

Ben Moon

Jack Hughes

Augusta University

Harley Eades III

Aubrey Bryant


Great PhD students!

## ALAN PERLIS AMERICAN COMPUTER SCIENTIST

A language that doesn't affect the way you think about programming, is not worth knowing.

CODE MESH LDN

# Data

~~Infinitely copiable~~

~~Arbitrarily discardable~~

~~Universally unconstrained~~

# Data

## as a resource

#CodeMeshLDN

# Linear types as a basis

1987

**LINEAR LOGIC***

Jean-Yves GIRARD

*Équipe de Logique Mathématique, UA 753 du CNRS, UER de Mathématiques, Université de Paris VII, 75251 Paris, France*

Communicated by M. Nivat
Received October 1986

*A la mémoire de Jean van Heijenoort*

**Abstract.** The familiar connective of negation is broken into two operations: linear negation which is the purely negative part of negation and the modality "of course" which has the meaning of a reaffirmation. Following this basic discovery, a completely new approach to the whole area between constructive logics and programmation is initiated.

1990

Linear types can change the world!

Philip Wadler
University of Glasgow*

Abstract

The linear logic of J.-Y. Girard suggests a new type system for functional languages, one which supports operations that "change the world". Values belonging to a linear type must be used exactly once: like the world, they cannot be duplicated or destroyed. Such values require no reference counting or garbage collection, and safely admit destructive array update. Linear types extend Schmidt's notion of single threading; provide an alternative to Hudak and Bloss' update analysis; and offer a practical complement to Lafont and Holmström's elegant linear languages.

An old canard against functional languages is that they cannot change the world:

$\Box$ modality — use any number of times

linear types — use exactly once

□ modality — use any number of times

□n modality — use n number of times

linear types — use exactly once

# The granule language

GADTs

Indexed types

+

Linear types

+

Graded modalities

Precision

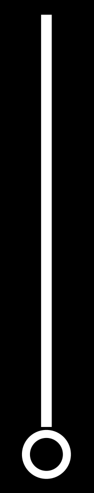
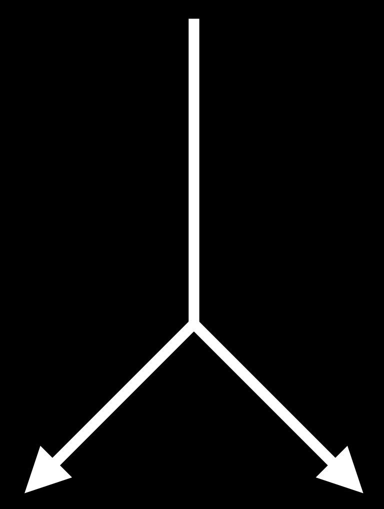
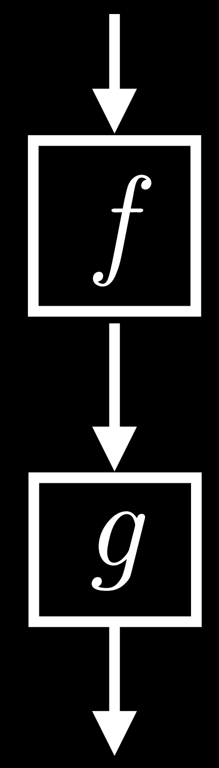
Data as resource

Quantitative reasoning

Discharge constraints automatically

SMT solver

# Graded modalities capture dataflow

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

Grading algebra (semiring) captures dataflow

# Graded modalities

## Two flavours

A [r]

**Graded "comonads"**
Backwards dataflow

A <r>

**Graded "monads"**
Forwards dataflow

| | |
|---|---|
| **Precision** | Indexed types |
| | + |
| **Data as resource** | Linear types |
| | + |
| **Quantitative reasoning** | Graded modalities |

=

# Fine-grained reasoning about "Data As Resource"

# Download and play!

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