

TLLA 2021

Linear Exponentials as Graded Modal Types

Jack Hughes

University of
Kent

Daniel Marshall

University of
Kent

James Wood

University of
Strathclyde
Glasgow

The logo of the University of Strathclyde Glasgow, featuring a shield with a white cross and four quadrants containing red, blue, and green symbols.

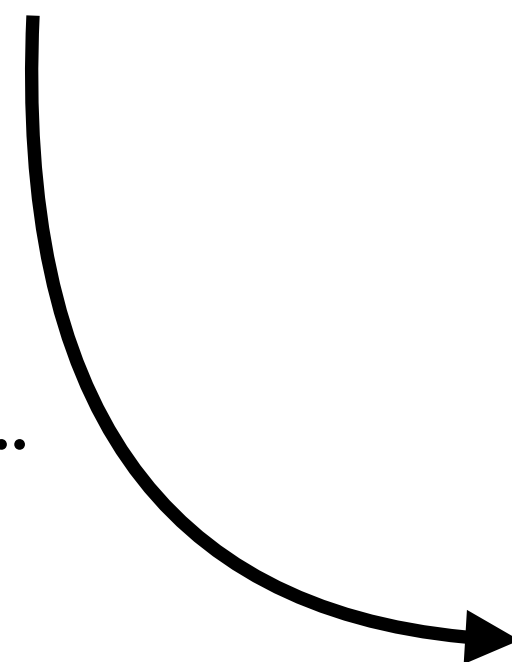
Dominic Orchard

University of
Kent

Linear Logic

! A modality represents non-linear usage of A

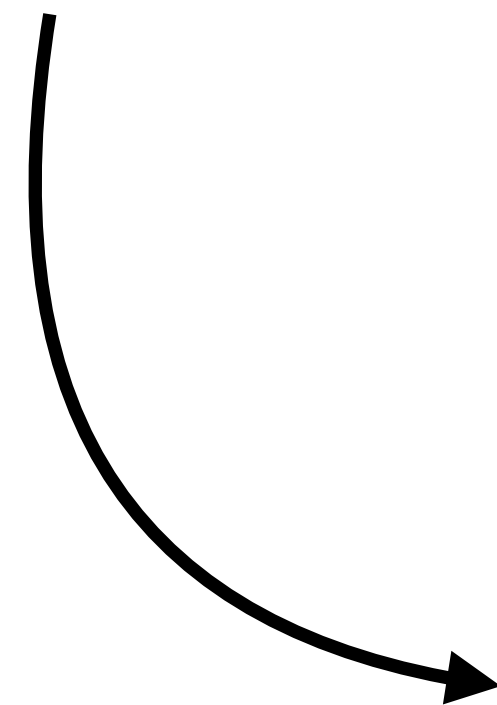
generalises to...



Bounded Linear Logic

! _{r} A family of modalities where r gives an upper bound on usage

generalises to...?



Graded Modal Types

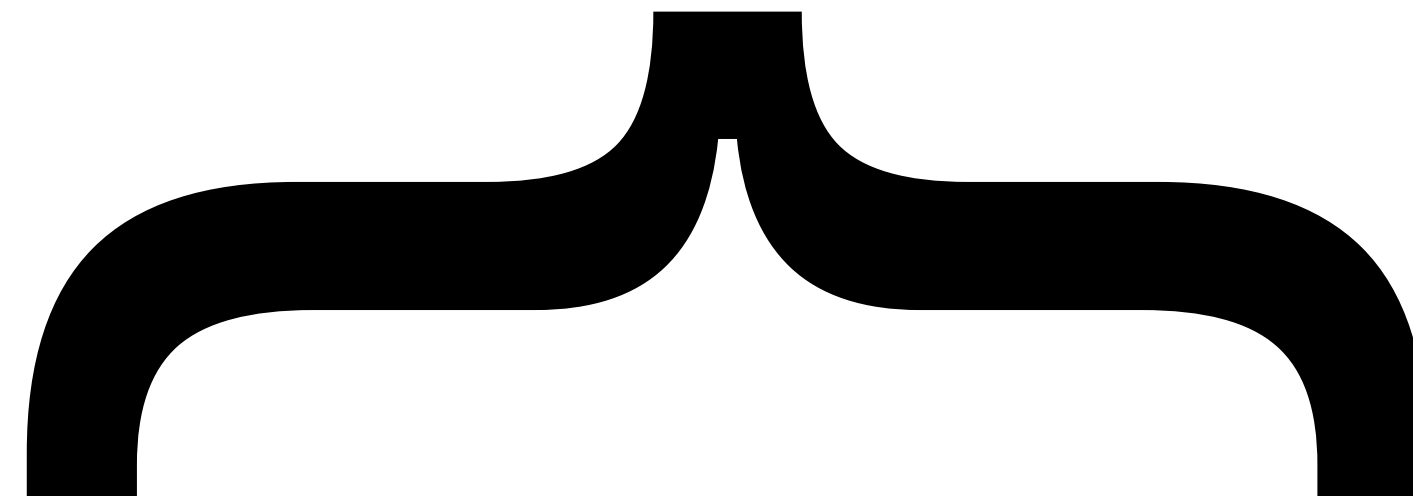
\square_r A family of modalities where r is drawn from a pre-ordered semiring $(\mathcal{R}, *, 1, +, 0, \sqsubseteq)$



Granule



 **Idris**



Linear Types

(data as a resource)

Graded Modal Types

(quantitative reasoning)

Indexed Types

(precision)

Demonstration

The Problem

In Granule... (and Linear Haskell, and Idris 2...)

```
push : forall {a b : Type, s : Semiring, r : s} . (a, b) [r] → (a [r], b [r])
push [(x, y)] = ([x], [y])
```

but in linear logic...

$$push_! : !(A \otimes B) \multimap !A \otimes !B$$

is **not** derivable!

The Solution

No semiring that can represent the ! modality...

...so we must augment our semiring with an additional operation.

$$\times : \mathcal{R} \times \mathcal{R} \rightarrow \mathcal{R}$$

(pronounced **hsup**)

Not too difficult to apply to other graded type systems!

The $\{0, 1, \omega\}$ Semiring

0 must be discarded

1 must be used linearly

ω permits unconstrained use

In Granule:

Zero

One

Many

We want to represent !A as $\square_{\omega} A$.

Pattern Matching

$$?r \vdash p : A \triangleright \Delta$$

before...

$$\frac{r \vdash p_1 : A \triangleright \Gamma_1 \quad s \vdash p_2 : B \triangleright \Gamma_2}{r \sqcup s \vdash (p_1, p_2) : A \otimes B \triangleright \Gamma_1, \Gamma_2} \text{ [PPROD]}$$

$$\frac{r \vdash p_1 : A \triangleright \Gamma_1 \quad s \vdash p_2 : B \triangleright \Gamma_2}{r \times s \vdash (p_1, p_2) : A \otimes B \triangleright \Gamma_1, \Gamma_2} \text{ [PPROD]}$$

...after

In semirings where we are okay with allowing push, just set $\times = \sqcup$!

The \times Operation

$$r \times s = \begin{cases} 1 & r = 1 \wedge s = 1 \\ \perp & \text{otherwise} \end{cases}$$

okay to push if both grades are 1!

$$(A \otimes B) \multimap A \otimes B$$

otherwise, pushing is forbidden.

$$\text{push}_! : !(A \otimes B) \multimap !A \otimes !B$$

Representing !

We want to represent !A as $\square_{\omega} A$.

So in Granule...

```
push : forall {a b : Type} . (a, b) [Many] → (a [Many], b [Many])  
push [(x, y)] = ([x], [y])
```

...should be disallowed.

Demonstration

The main theorem:

The adjusted Granule core calculus, for the $\{0, 1, \omega\}$ semiring with $!A = \square_{\omega} A$, has the same expressive power as IMELL.

References

For more on Granule and graded types...

[Quantitative program reasoning with graded modal types](#)

Dominic Orchard, Vilem Benjamin-Liepelt, Harley Eades III (ICFP 2019)

For more on distributive laws...

[Deriving distributive laws for graded linear types](#)

Jack Hughes, Michael Vollmer, Dominic Orchard (LINEARITY/TLLA 2020)

And read the rest of this paper for more details!

[Linear exponentials as graded modal types](#)

Jack Hughes, Daniel Marshall, James Wood, Dominic Orchard (TLLA 2021)

Thank you!

Publications, implementation and more at:
granule-project.github.io

Find us on Twitter at:
[@granulelang](https://twitter.com/granulelang)

Find **me** on Twitter at
[@starsandspirals](https://twitter.com/starsandspirals)  