Replicate, Reuse, Repeat:

Sunday 3rd April
PLACES 2022

capturing *non-linear* communication via **session types** and **graded modal types**

Daniel Marshall
Dominic Orchard

University of Kent
gr

The Granule programming language

Linear types

Indexed types

Graded modal types

Linear types are like **cakes**.

You can only eat them **once**.

You **have** to eat them.


```
desire : Cake → (Happy, Cake)
desire cake = (eat cake, have cake)
```

Granule's session types

(based on the GV calculus)

```
send          :  LChan (Send a p) → a → LChan p
recv          :  LChan (Recv a p) → (a, LChan p)
forkLinear    :  (LChan p → ())   → LChan (Dual p)
close         :  LChan End → ()

selectLeft    :  LChan (Select p1 p2) → LChan p1
selectRight   :  LChan (Select p1 p2) → LChan p2
offer         :  (LChan p1 → a) → (LChan p2 → a)
                 → LChan (Offer p1 p2) → a
```

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

$\Box_r A$ family of modalities where r is drawn from a **pre-ordered semiring**

Graded modal types in action!

Reusable channels

```
forkLinear:
    ⇒ (LChan p → ())
    → LChan (Dual p)
```

```
forkNonLinear : {SingleAction p}
    ⇒ ((LChan p) [r] → ())
    → (LChan (Dual p)) [r]
```

`{SingleAction p}`

restricts to only protocols with a single action, such as

```
Send a End
Recv a End
```

`forkNonLinear`

allows for channels with **arbitrary** grades – for example, channels that can be shared and **reused**!

Replicated servers

```
forkReplicate : {ReceivePrefix p}
    ⇒ (LChan p → ()) [0..n]
    → N n
    → Vec n ((LChan (Dual p)) [0..1])
```

`{ReceivePrefix p}`

restricts to only protocols with a prefix that receives, such as

```
Recv a p
Offer p1 p2
```

`forkReplicate`

allows for **one** graded **server** to be interacted with by a vector of **multiple** dual clients!

Multicast sending

```
forkMulticast : {Sends p}
    ⇒ (LChan (Graded n p) → ())
    → N n
    → Vec n (LChan (Dual p))
```

`{Sends p}`

restricts to only protocols where every step is a send, such as

```
Send a (Send a End)
Select p1 p2
```

`forkMulticast`

allows for **one** graded **value** to be sent, and then received by a vector of **multiple** channels!

...but why can't we just **promote?**

In older versions of Granule, channels could be promoted - **but**:

problems arise in **call-by-value** settings (like Granule, by default!)

Solution: **restrict** promotion when creating linear channels!

(and **unique arrays** - see our ESOP talk for more!)

Thanks for listening!

In summary:

Leveraged **session types** alongside **graded modal types**

Captured **non-linear** behaviour in a controlled way

Introduced a suite of primitives for **reuse, replication** and **repetition**

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