Towards Quantitative Inductive Families

Yulong Huang & Jeremy Yallop

University of Cambridge, UK

TYPES, IT University of Copenhagen

June 2024

Quantitative type theory (QTT)

- Linear (substructural) types + dependent types
- Runtime irrelevance in Agda
- Multiplicity in Idris2

```
append :
  (1 _ : Vect n a) ->
  (1 _ : Vect m a) ->
  Vect (n + m) a
append xs ys = ?append_rhs

Main >:t append_rhs
  0 m : Nat
  0 a : Type
  0 n : Nat
  1 ys : Vect m a
  1 xs : Vect n a
--------------------------------------
append_rhs : Vect (plus n m) a
```

Quantitative type theory + Datatypes?

Indexed datatypes → Inductive families

Indexed datatypes with quantities → ? ? ?

Quantitative type theory + Datatypes?

$$\frac{\gamma \blacktriangleright z \qquad \delta, p, r \blacktriangleright s \qquad \eta \blacktriangleright n \qquad \theta, q \blacktriangleright A}{(\gamma \wedge \eta) \circledast_r (\delta + p\eta) \blacktriangleright \mathsf{natrec}^{q}_{p,r}\; A\; z\; s\; n}$$

[1] Abel et al.

$$\frac{\begin{array}{l}(\Delta \mid \sigma_3 \mid \sigma_1 + \sigma_2) \odot \Gamma \vdash t_1 : (x :_r A) \otimes B \\ (\Delta, (\sigma_1 + \sigma_2) \mid \sigma_5, r' \mid \mathbf{0}) \odot \Gamma, z : (x :_r A) \otimes B \vdash C : \mathsf{Type}_l \\ (\Delta, \sigma_1, (\sigma_2, r) \mid \sigma_4, s, s \mid \sigma_5, r', r') \odot \Gamma, x : A, y : B \vdash t_2 : [(x, y)/z]C\end{array}}{(\Delta \mid \sigma_4 + s * \sigma_3 \mid \sigma_5 + r' * \sigma_3) \odot \Gamma \vdash \mathsf{let}\,(x, y) = t_1 \,\mathsf{in}\, t_2 : [t_1/z]C} \otimes_e$$

[2] Moon et al.

$$\frac{\begin{array}{l}0\Gamma, x \overset{0}{:} \mathsf{Nat} \vdash P\ \mathsf{type} \\ \Gamma \vdash M \overset{\sigma}{:} \mathsf{Nat} \\ 0\Gamma \vdash N_z \overset{\sigma}{:} P[\mathsf{zero}/x] \\ 0\Gamma, n \overset{0}{:} \mathsf{Nat}, p \overset{\sigma}{:} P[n/x] \vdash N_s \overset{\sigma}{:} P[\mathsf{succ}(n)/x]\end{array}}{\Gamma \vdash \mathsf{rec}_{x.P}\, M\, \{\mathsf{zero} \mapsto N_z; \mathsf{succ}(n; p) \mapsto N_s\} \overset{\sigma}{:} P[M/x]}$$

[3] Atkey

T-SIGMAELIM

$$\frac{\begin{array}{c}\Delta\, ;\, \Gamma_1 \vdash a : \Sigma x{:}^{q} A_1.A_2 \\ \Delta, x{:}A_1, y{:}A_2\, ;\, \Gamma_2, x :^{q} A_1, y :^{1} A_2 \vdash b : B\{(x, y)/z\} \\ \Delta, z{:}(\Sigma x{:}^{q} A_1.A_2)\, ;\, \Gamma_3, z :^{r} (\Sigma x{:}^{q} A_1.A_2) \vdash B : \mathbf{type}\end{array}}{\Delta\, ;\, \Gamma_1 + \Gamma_2 \vdash \mathbf{let}\,(x, y) = a\, \mathsf{in}\, b : B\{a/z\}}$$

[4] Choudhury et al.

Motivation

- Have a theoretical foundation of datatypes in QTT
- Have a unified framework for future research
- Investigate problems like pattern matching elimination

What we want from datatypes

- Expressive and intuitive

  Pairs, vectors, trees (etc.) with different usages for each component

- Compiler-friendly

  Small effort to type-check, easy to implement

- Syntactically well-behaved

  Substitution, reduction, erasure, ...

- Semantically meaningful

  Initial algebra semantics

Structure of this talk

I. Typing rules and principles of QTT

- Domain of quantities
- Types need nothing
- No erased at runtime
- Join over branches

II. Extending QTT with datatypes

- Linear Lists
- Vectors with quantities

I. Typing rules and principles of QTT

Zero, One, and Many

Each variable is assigned with a quantity from $Q = \{0, 1, \omega\}$, marking its runtime usage.

Quantity of a variable	**Meaning**
0	unused
1	used linearly
ω	used non-linearly

Zero, One, and Many

We can describe more situations using an order and a few operations over the quantities.

Order: $0 \leqslant \omega \geqslant 1$.

Quantity of a variable	**Meaning**
$p + q$	used p times in an expression, and then q times in another expression
$p \cdot q$	used p times in an expression, and that expression is used q times
$p \sqcup q$	used p or q times, taking the least upper bound of p and q

QTT judgements

$$\Gamma \vdash M \overset{\sigma}{:} A \;;\; \underline{m}$$

σ : type-checking modes from $\{\; \underset{\text{erased}}{0} \;,\; \underset{\text{runtime}}{1} \;\}$

$\underline{m} : dom(\Gamma) \longrightarrow Q$, where $\underline{m}(x)$ is runtime usage of x

The variable rule

$$\frac{}{x{:}A, y{:}B \vdash x \overset{0}{:} A \;;\; 0, 0}$$

$$\frac{}{x{:}A, y{:}B \vdash x \overset{1}{:} A \;;\; 1, 0}$$

TY-VAR

$$\frac{x : A \in \Gamma}{\Gamma \vdash x \overset{\sigma}{:} A \;;\; \sigma_x}$$

$$\begin{cases} \sigma_x(x) = \sigma \\ \sigma_x(y) = 0 \end{cases}$$

Note: we abuse the notation and implicitly cast modes to quantities here.

Principle I: Types need nothing

$$\textsc{ty-Pi}\quad \frac{\Gamma \vdash A \overset{0}{:} \mathrm{Type} \;;\; \underline{\mathit{0}} \qquad \Gamma, x{:}A \vdash B \overset{0}{:} \mathrm{Type} \;;\; \underline{\mathit{0}}}{\Gamma \vdash \Pi x \overset{q}{:} A.\, B \overset{0}{:} \mathrm{Type} \;;\; \underline{\mathit{0}}}$$

Type-formers like Π are judged in the erased mode.

Lambda and application

$$\textsc{ty-Lam}\quad \frac{\Gamma, x{:}A \vdash M \overset{\sigma}{:} B \;;\; \underline{m}, q}{\Gamma \vdash \lambda x \overset{q}{:} A.\, M \overset{\sigma}{:} \Pi x \overset{q}{:} A.\, B \;;\; \underline{m}}$$

If the fresh variable x is used q times,

then we get a function of type $\Pi x \overset{q}{:} A.\ B$.

$$\textsc{ty-App}\quad \frac{\sigma' = 0 \Leftrightarrow (\sigma = 0 \vee q = 0) \qquad \Gamma \vdash M \overset{\sigma}{:} \Pi x \overset{q}{:} A.\, B \;;\; \underline{m} \qquad \Gamma \vdash N \overset{\sigma'}{:} A \;;\; \underline{n}}{\Gamma \vdash M\ N \overset{\sigma}{:} B[M/x] \;;\; \underline{m} + q\underline{n}}$$

Variables are used $\underline{m}$ times in M and $\underline{n}$ times in N.

So, the total usage is $\underline{m} + q\underline{n}$,

since N is used q times by function M.

Note: operations on Q extend pointwise to quantity assignments.

Principle II: No erased at runtime

TY-LAM

$$\frac{\Gamma, x : A \vdash M \overset{\sigma}{:} B \;;\; \underline{m}, q}{\Gamma \vdash \lambda x \overset{q}{:} A.\, M \overset{\sigma}{:} \Pi x \overset{q}{:} A.\, B \;;\; \underline{m}}$$

TY-APP

$$\frac{\sigma' = 0 \Leftrightarrow (\sigma = 0 \lor q = 0) \qquad \Gamma \vdash M \overset{\sigma}{:} \Pi x \overset{q}{:} A.\, B \;;\; \underline{m} \qquad \Gamma \vdash N \overset{\sigma'}{:} A \;;\; \underline{n}}{\Gamma \vdash M\ N \overset{\sigma}{:} B[M/x] \;;\; \underline{m} + q\underline{n}}$$

N is erased $(\sigma' = 0)$ iff:

(1). The application is erased $(\sigma' = 0)$.

(2). M doesn't use its argument $(q = 0)$.

Principle III: Join over branches

TY-ELIMBOOL

$$\frac{\begin{array}{c}\Gamma, b{:}\mathbf{Bool} \vdash P \overset{0}{:} \mathrm{Type} \ ; \ \underline{0} \\ \Gamma \vdash L \overset{\sigma}{:} \mathbf{Bool} \ ; \ \underline{l} \\ \Gamma \vdash M \overset{\sigma}{:} P[\mathbf{tt}/b] \ ; \ \underline{m} \\ \Gamma \vdash N \overset{\sigma}{:} P[\mathbf{ff}/b] \ ; \ \underline{n}\end{array}}{\Gamma \vdash \mathbf{if}_P \ L \ \mathbf{then} \ M \ \mathbf{else} \ N \overset{\sigma}{:} P[L/b] \ ; \ \underline{l} + (\underline{m} \sqcup \underline{n})}$$

Order: $0 \leqslant \omega \geqslant 1$.

$p \sqcup q$: the variable is used p or q times,

taking the least upper bound of p and q.

Variables are used $\underline{l}$ times in L,
and $(\underline{m} \sqcup \underline{n})$ times in the branches.

So, the total usages are $\underline{l} + (\underline{m} \sqcup \underline{n})$.

Sub-usaging

TY-SUB-USAGE

$$\frac{\Gamma \vdash M \overset{\sigma}{:} A \;;\; \underline{m} \qquad \underline{m} \leq \underline{m'}}{\Gamma \vdash M \overset{\sigma}{:} A \;;\; \underline{m'}}$$

We can over-estimate the resources required by M.

Syntactic properties

Lemma (Substitution). The following rule for substitution is admissible:

TY-SUBST

$$\frac{\Gamma, x{:}A \vdash M \overset{\sigma}{:} B \;;\; \underline{m}, q \qquad \Gamma \vdash N \overset{\sigma'}{:} A \;;\; \underline{n} \qquad \sigma' = 0 \Leftrightarrow q = 0}{\Gamma \vdash M[N/x] \overset{\sigma}{:} B[N/x] \;;\; \underline{m} + q\underline{n}}$$

Similar to the application rule.

Lemma (Reduction). If $\Gamma \vdash M \overset{\sigma}{:} A \;;\; \underline{m}$ and M reduces to M', then $\Gamma \vdash M' \overset{\sigma}{:} A \;;\; \underline{m}$ is derivable.

II. Extending QTT with datatypes

What does "using" mean?

Using means matching

Using M once is matching/unfolding it once.

For example, if M is a pair, then

$$\text{let } (x, y) = M \ \text{ in } \ N$$

counts as using M once.

Usage of M's components are specified by the type of M.

Linear lists: signature and type former

$$\texttt{data}\ \mathbf{List}^{11}\ (A{:}\mathrm{Type}) : \mathrm{Type}\ \texttt{where}$$
$$[\,] : \mathbf{List}^{11} A$$
$$\_{::}\_ : (x \overset{1}{:} A)(xs \overset{1}{:} \mathbf{List}^{11} A) \to \mathbf{List}^{11} A$$

Head and tail can be used only once.

$$\textsc{ty-List}\ \frac{\Gamma \vdash A \overset{0}{:} \mathrm{Type} \,;\ \underline{0}}{\Gamma \vdash \mathbf{List}^{11} A \overset{0}{:} \mathrm{Type} \,;\ \underline{0}}$$

Principle I: Types need nothing.

It's easy to extend this to vectors, we'll focus on lists for now!

Linear lists: constructors

TY-NIL

$$\frac{\Gamma \vdash A \overset{0}{:} \text{Type} \; ; \; \underline{\mathit{0}}}{\Gamma \vdash [\,] \overset{\sigma}{:} \textbf{List}^{11} A \; ; \; \underline{\mathit{0}}}$$

No resource required for Nil.

TY-CONS

$$\frac{\Gamma \vdash M \overset{\sigma}{:} A \; ; \; \underline{m} \qquad \Gamma \vdash N \overset{\sigma}{:} \textbf{List}^{11} A \; ; \; \underline{n}}{\Gamma \vdash M :: N \overset{\sigma}{:} \textbf{List}^{11} A \; ; \; \underline{m} + \underline{n}}$$

Resources are summed up in constructors, like what we did in applications.

Linear lists: eliminator

TY-ELIMLIST

$$\frac{\begin{array}{c}\Gamma, ls : \mathbf{List}^{11} A \vdash P \overset{0}{:} \text{Type} \ ; \ \underline{0} \\ \Gamma \vdash L \overset{\sigma}{:} \mathbf{List}^{11} A \ ; \ \underline{l} \\ \Gamma \vdash M \overset{\sigma}{:} P[[\,]/ls] \ ; \ \underline{m} \\ \Gamma, x : A, xs : \mathbf{List}^{11} A, r : P[xs/ls] \vdash N \overset{\sigma}{:} P[x :: xs/ls] \ ; \ \underline{n}, 1, 0, 1\end{array}}{\Gamma \vdash Elim_{list}(P, L, M, (x, xs, r).N) \overset{\sigma}{:} P[L/ls] \ ; \ \underline{l} + \underline{m} + \omega \underline{n}}$$

It looks messy...

Linear lists: eliminator

TY-ELIMLIST

$$\frac{\begin{array}{c}\Gamma, ls{:}\mathbf{List}^{11} A \vdash P \overset{0}{:} \mathrm{Type} \;;\; \underline{0} \\ \Gamma \vdash L \overset{\sigma}{:} \mathbf{List}^{11} A \;;\; \underline{l} \\ \Gamma \vdash M \overset{\sigma}{:} P[[\,]/ls] \;;\; \underline{m} \\ \Gamma, x{:}A, xs{:}\mathbf{List}^{11} A, r{:}P[xs/ls] \vdash N \overset{\sigma}{:} P[x :: xs/ls] \;;\; \underline{n}, 1, 0, 1\end{array}}{\Gamma \vdash Elim_{list}(P, L, M, (x, xs, r).N) \overset{\sigma}{:} P[L/ls] \;;\; \underline{l} + \underline{m} + \omega\underline{n}}$$

It looks messy…

but it's just about typing and quantities.

Linear lists: eliminator

TY-ELIMLIST

$$\frac{\begin{array}{c}\Gamma, ls\!:\!\mathbf{List}^{11} A \vdash P \overset{0}{:} \mathrm{Type} \;;\; \underline{\mathit{0}} \\ \Gamma \vdash L \overset{\sigma}{:} \mathbf{List}^{11} A \;;\; \underline{l} \\ \Gamma \vdash M \overset{\sigma}{:} P[[]/ls] \;;\; \underline{m} \\ \Gamma, x\!:\!A, xs\!:\!\mathbf{List}^{11} A, r\!:\!P[xs/ls] \vdash N \overset{\sigma}{:} P[x :: xs/ls] \;;\; \underline{n}, 1, 0, 1\end{array}}{\Gamma \vdash \mathit{Elim}_{list}(P, L, M, (x, xs, r).N) \overset{\sigma}{:} P[L/ls] \;;\; \underline{l} + \underline{m} + \omega \underline{n}}$$

The typing rules are conventional.

Linear lists: eliminator

$$\frac{\begin{array}{c}\Gamma, ls{:}\mathbf{List}^{11} A \vdash P \overset{0}{:} \text{Type} \; ; \; \underline{0} \\ \Gamma \vdash L \overset{\sigma}{:} \mathbf{List}^{11} A \; ; \; \underline{l} \\ \Gamma \vdash M \overset{\sigma}{:} P[[\,]/ls] \; ; \; \underline{m} \\ \Gamma, x{:}A, xs{:}\mathbf{List}^{11} A, r{:}P[xs/ls] \vdash N \overset{\sigma}{:} P[x :: xs/ls] \; ; \; \underline{n}, 1, 0, 1\end{array}}{\Gamma \vdash Elim_{list}(P, L, M, (x, xs, r).N) \overset{\sigma}{:} P[L/ls] \; ; \; \underline{l} + \underline{m} + \omega \underline{n}} \;\text{TY-ELIMLIST}$$

The typing rules are conventional.

We focus on quantities.

Linear lists: eliminator

TY-ELIMLIST

$$\frac{\begin{array}{c}\Gamma, ls{:}\mathbf{List}^{11} A \vdash P \overset{0}{:} \text{Type} \; ; \; \underline{0} \\ \Gamma \vdash L \overset{\sigma}{:} \mathbf{List}^{11} A \; ; \; \underline{l} \\ \Gamma \vdash M \overset{\sigma}{:} P[[\,]/ls] \; ; \; \underline{m} \\ \Gamma, x{:}A, xs{:}\mathbf{List}^{11} A, r{:}P[xs/ls] \vdash N \overset{\sigma}{:} P[x :: xs/ls] \; ; \; \underline{n}, 1, 0, 1\end{array}}{\Gamma \vdash Elim_{list}(P, L, M, (x, xs, r).N) \overset{\sigma}{:} P[L/ls] \; ; \; \underline{l} + \underline{m} + \omega\underline{n}}$$

P is a type that needs nothing.

Variables are used $\underline{l}$ times in L and $\underline{m}$ times in the Nil case M.

Linear lists: eliminator

TY-ELIMLIST

$$\frac{\begin{array}{c}\Gamma, ls : \mathbf{List}^{11} A \vdash P \overset{0}{:} \text{Type} \; ; \; \underline{0} \\ \Gamma \vdash L \overset{\sigma}{:} \mathbf{List}^{11} A \; ; \; \underline{l} \\ \Gamma \vdash M \overset{\sigma}{:} P[[\,]/ls] \; ; \; \underline{m} \\ \Gamma, x : A, xs : \mathbf{List}^{11} A, r : P[xs/ls] \vdash N \overset{\sigma}{:} P[x :: xs/ls] \; ; \; \underline{n}, \overset{x}{1}, \overset{xs}{0}, \overset{r}{1}\end{array}}{\Gamma \vdash Elim_{list}(P, L, M, (x, xs, r).N) \overset{\sigma}{:} P[L/ls] \; ; \; \underline{l} + \underline{m} + \omega \underline{n}}$$

At the Cons case:

Usages of variables in the context is $\underline{n}$.

Head x and the induction hypothesis r are used once.

The tail xs is only there for typing, hence it has usage 0.

Linear lists: eliminator

TY-ELIMLIST

$$\frac{\begin{array}{c}\Gamma, ls{:}\textbf{List}^{11} A \vdash P \overset{0}{:} \text{Type} \; ; \; \underline{0} \\ \Gamma \vdash L \overset{\sigma}{:} \textbf{List}^{11} A \; ; \; \underline{l} \\ \Gamma \vdash M \overset{\sigma}{:} P[[\,]/ls] \; ; \; \underline{m} \\ \Gamma, x{:}A, xs{:}\textbf{List}^{11} A, r{:}P[xs/ls] \vdash N \overset{\sigma}{:} P[x :: xs/ls] \; ; \; \underline{n}, 1, 0, 1\end{array}}{\Gamma \vdash Elim_{list}(P, L, M, (x, xs, r).N) \overset{\sigma}{:} P[L/ls] \; ; \; \underline{l} + \underline{m} + \omega\underline{n}}$$

If $L = [\,]$:

The eliminator reduces to M that requires resources $\underline{m}$.

If $L = Hd :: Tl$:

N is evaluated many times until it hits the base case M, using resources $\underline{m} + \omega\underline{n}$ (the exact number of iterations is unknown).

Linear lists: eliminator

TY-ELIMLIST

$$\frac{\begin{array}{c}\Gamma, ls{:}\mathbf{List}^{11} A \vdash P \overset{0}{:} \text{Type} \;;\; \underline{0} \\ \Gamma \vdash L \overset{\sigma}{:} \mathbf{List}^{11} A \;;\; \underline{l} \\ \Gamma \vdash M \overset{\sigma}{:} P[[\,]/ls] \;;\; \underline{m} \\ \Gamma, x{:}A, xs{:}\mathbf{List}^{11} A, r{:}P[xs/ls] \vdash N \overset{\sigma}{:} P[x :: xs/ls] \;;\; \underline{n}, 1, 0, 1\end{array}}{\Gamma \vdash \mathit{Elim}_{list}(P, L, M, (x, xs, r).N) \overset{\sigma}{:} P[L/ls] \;;\; \underline{l} + \underline{m} + \omega\underline{n}}$$

Principle III: Join over branches.

We join the usages of two cases, and add the usages in the list L, getting $\underline{l} + (\underline{m} \sqcup (\underline{m} + \omega\underline{n}))$, which simplifies to $\underline{l} + (\underline{m} + \omega\underline{n})$.

Use it now, or use it later

Sometimes, we want to use the tail directly instead of recursively.

TY-ELIMLIST

$$\frac{\begin{array}{c}\Gamma, ls{:}\mathbf{List}^{11} A \vdash P \overset{0}{:} \text{Type} \;;\; \underline{0} \\ \Gamma \vdash L \overset{\sigma}{:} \mathbf{List}^{11} A \;;\; \underline{l} \\ \Gamma \vdash M \overset{\sigma}{:} P[[\,]/ls] \;;\; \underline{m} \\ \Gamma, x{:}A, xs{:}\mathbf{List}^{11} A, r{:}P[xs/ls] \vdash N \overset{\sigma}{:} P[x :: xs/ls] \;;\; \underline{n}, \overset{x}{1}, \overset{xs}{1}, \overset{r}{0}\end{array}}{\Gamma \vdash Elim_{list}(P, L, M, (x, xs, r).N) \overset{\sigma}{:} P[L/ls] \;;\; \underline{l} + (\underline{m} \sqcup \underline{n})}$$

At the Cons case:

Usages of variables in the context is $\underline{n}$.

Head x and tail xs are each used once.

The induction hypothesis r is unused.

Use it now, or use it later

TY-ELIMLIST

$$\frac{\begin{array}{c}\Gamma, ls{:}\mathbf{List}^{11} A \vdash P \overset{0}{:} \text{Type} \; ; \; \underline{0} \\ \Gamma \vdash L \overset{\sigma}{:} \mathbf{List}^{11} A \; ; \; \underline{l} \\ \Gamma \vdash M \overset{\sigma}{:} P[[\,]/ls] \; ; \; \underline{m} \\ \Gamma, x{:}A, xs{:}\mathbf{List}^{11} A, r{:}P[xs/ls] \vdash N \overset{\sigma}{:} P[x :: xs/ls] \; ; \; \underline{n}, 1, 1, 0\end{array}}{\Gamma \vdash Elim_{list}(P, L, M, (x, xs, r).N) \overset{\sigma}{:} P[L/ls] \; ; \; \underline{l} + (\underline{m} \sqcup \underline{n})}$$

So, the total usages are $\underline{l} + (\underline{m} \sqcup \underline{n})$,

similar to Bool.

Putting it together

TY-ELIMLIST2

$$\frac{\begin{array}{c}\Gamma, ls : \mathbf{List}^{11} A \vdash P \overset{0}{:} \text{Type} \;;\; \underline{\mathit{0}} \\ \Gamma \vdash L \overset{\sigma}{:} \mathbf{List}^{11} A \;;\; \underline{l} \\ \Gamma \vdash M \overset{\sigma}{:} P[[\,]/ls] \;;\; \underline{m} \\ \Gamma, x : A, xs : \mathbf{List}^{11} A, r : P[xs/ls] \vdash N \overset{\sigma}{:} P[x :: xs/ls] \;;\; \underline{n}, 1, q_1, q_2 \\ q_1 + q_2 \; \leq \; 1\end{array}}{\Gamma \vdash Elim_{list}(P, L, M, (x, xs, r).N) \overset{\sigma}{:} P[L/ls] \;;\; \underline{l} + (\underline{m} \sqcup (q_2 \underline{m} + (q_2 + 1)\underline{n}))}$$

Use it now, or use it later

The combined usage can't be greater

(than 1)

Note: recall that $0 \leqslant \omega \geqslant 1$, so (q_1, q_2) can only be (0,1) or (1,0).

Putting it together

TY-ElimList2

$$\frac{\begin{array}{c}\Gamma, ls{:}\mathbf{List}^{11} A \vdash P \overset{0}{:} \text{Type} \;;\; \underline{0} \\ \Gamma \vdash L \overset{\sigma}{:} \mathbf{List}^{11} A \;;\; \underline{l} \\ \Gamma \vdash M \overset{\sigma}{:} P[[\,]/ls] \;;\; \underline{m} \\ \Gamma, x{:}A, xs{:}\mathbf{List}^{11} A, r{:}P[xs/ls] \vdash N \overset{\sigma}{:} P[x :: xs/ls] \;;\; \underline{n}, 1, q_1, q_2 \\ q_1 + q_2 \le 1\end{array}}{\Gamma \vdash Elim_{list}(P, L, M, (x, xs, r).N) \overset{\sigma}{:} P[L/ls] \;;\; \underline{l} + (\underline{m} \sqcup (q_2\underline{m} + (q_2 + 1)\underline{n}))}$$

The formula covers all situations of q_2.

$q_2 = 0$: $\underline{l} + (\underline{m} \sqcup \underline{n})$

$q_2 = 1$: $\underline{l} + (\underline{m} \sqcup (\underline{m} + \omega\underline{n}))$

A glance at Vectors

$$\texttt{data}\ \mathbf{Vec}^{pq}\ (A{:}\mathrm{Type}) : (n : \mathbf{Nat}) \to \mathrm{Type}\ \texttt{where}$$
$$[\,] : \mathbf{Vec}^{pq} A\ 0$$
$$\_{::}\_ : (n \overset{0}{:} \mathbf{Nat})(x \overset{p}{:} A)(xs \overset{q}{:} \mathbf{Vec}^{pq} A\ n) \to \mathbf{Vec}^{pq} A\ s(n)$$

Vector size n is erased.

Head and tail can be used p and q times each.

TY-VEC-PQ

$$\frac{\Gamma \vdash A \overset{0}{:} \mathrm{Type}\ ;\ \underline{0} \qquad \Gamma \vdash n \overset{0}{:} \mathbf{Nat}\ ;\ \underline{0}}{\Gamma \vdash \mathbf{Vec}^{pq} A\ n \overset{0}{:} \mathrm{Type}\ ;\ \underline{0}}$$

Types need nothing.

Note: it means that one could define a datatype Vec^{pq} for all (p, q) in Q.

Note: we don't specify quantities for parameters and indices because they cannot be "used" at runtime.

Vectors: constructors

TY-VCONS-PQ

$$\frac{\begin{array}{c} \sigma_1 = 0 \Leftrightarrow (\sigma = 0 \lor p = 0) \\ \sigma_2 = 0 \Leftrightarrow (\sigma = 0 \lor q = 0) \\ \Gamma \vdash n \overset{0}{:} \mathbf{Nat} \; ; \; \underline{0} \\ \Gamma \vdash M \overset{\sigma_1}{:} A \; ; \; \underline{m} \\ \Gamma \vdash N \overset{\sigma_2}{:} \mathbf{Vec}^{pq} A \; n \; ; \; \underline{n} \end{array}}{\Gamma \vdash M :: N \overset{\sigma}{:} \mathbf{Vec}^{pq} A \; s(n) \; ; \; p\underline{m} + q\underline{n}}$$

Principle II: No erased at runtime.
The constraints make sure there is no erased term at runtime.

Resources are summed up like what we did in applications.

Vectors: eliminators

TY-ElimVec-pq

$$\frac{\begin{array}{c}\Gamma, n{:}\mathbf{Nat}, ls{:}\mathbf{Vec}^{pq} A\ n \vdash P \overset{0}{:} \text{Type} \ ;\ \underline{0} \\ \Gamma \vdash L \overset{\sigma}{:} \mathbf{Vec}^{pq} A\ n \ ;\ \underline{l} \\ \Gamma \vdash M \overset{\sigma}{:} P[0/n, []/ls] \ ;\ \underline{m} \\ \Gamma, n{:}\mathbf{Nat}, x{:}A, xs{:}\mathbf{List}^{pq} A, r{:}P[n/n, xs/ls] \vdash N \overset{\sigma}{:} P[s(n)/n, x :: xs/ls] \ ;\ \underline{n}, 0, p, q_1, q_2 \\ q_1 + q_2 \leq q\end{array}}{\Gamma \vdash Elim_{list}(P, L, M, (n, x, xs, r).N) \overset{\sigma}{:} P[n/n, L/ls] \ ;\ \underline{l} + (\underline{m} \sqcup (q_2\underline{m} + (q_2 + 1)\underline{n}))}$$

TL;DR: It's basically the same as the list eliminator.

Now, and next

Progress so far:

- A general typing scheme for inductive-family definitions with quantities
- Proof of substitution and reduction for the general scheme (on paper)
- A prototype implementation of the type checker in OCaml

Future work:

- Resource-aware operational semantics and theorems (erasure, linearity, etc.)
- Pattern matching elimination (internal in QTT)
- Initial algebra semantics and formalization

Thank you!

...and questions?

Speaker email: yh419@cam.ac.uk

References

[1] Andreas Abel, Nils Anders Danielsson, and Oskar Eriksson. A graded modal dependent type theory with a universe and erasure, formalized. Proc. ACM Program. Lang., 7(ICFP):920–954, 2023.

[2] Benjamin Moon, Harley Eades III, and Dominic Orchard. Graded modal dependent type theory. In European Symposium on Programming, pages 462–490. Springer International Publishing Cham, 2021.

[3] Robert Atkey. Polynomial time and dependent types. Proc. ACM Program. Lang., 8(POPL):2288– 2317, 2024.

[4] Pritam Choudhury, Harley Eades III, Richard A. Eisenberg, and Stephanie Weirich. A graded dependent type system with a usage-aware semantics. Proc. ACM Program. Lang., 5(POPL):1–32, 2021.