Quantitative polynomial functors

Georgi Nakov Fredrik Nordvall Forsberg

Department of Computer and Information Sciences

University of Strathclyde

CALCO 2021

September 17, 2021

Outline

Motivation

1. Investigate a systematic way of adding datatypes to dependent type theories in presence of linearity.
    - Our approach is based on Containers (Abbott, Altenkirch, and Ghani 2003).

Motivation

1. Investigate a systematic way of adding datatypes to dependent type theories in presence of linearity.
    - Our approach is based on Containers (Abbott, Altenkirch, and Ghani 2003).
2. Provide a formal theory of datatypes in Idris 2 (Brady 2021), Granule (Orchard, Liepelt, and Eades III 2019).

Linear logic and type theory

Adding linearity to dependent type theories:

Linear logic and type theory

Adding linearity to dependent type theories:

- allows for precise usage tracking, viewing data as a computational resource.

Linear logic and type theory

Adding linearity to dependent type theories:

- allows for precise usage tracking, viewing data as a computational resource.
- could enforce some restricted behaviour, e.g. in-place updates.

Linear logic and type theory

Adding linearity to dependent type theories:

- allows for precise usage tracking, viewing data as a computational resource.
- could enforce some restricted behaviour, e.g. in-place updates.
- could lead to more efficient implementations, e.g. erasability.

Linear logic and type theory

Adding linearity to dependent type theories:

- allows for precise usage tracking, viewing data as a computational resource.
- could enforce some restricted behaviour, e.g. in-place updates.
- could lead to more efficient implementations, e.g. erasability.

Active research area - Linear Logical Framework by Cervesato and Pfenning 2002,

Linear logic and type theory

Adding linearity to dependent type theories:

- allows for precise usage tracking, viewing data as a computational resource.
- could enforce some restricted behaviour, e.g. in-place updates.
- could lead to more efficient implementations, e.g. erasability.

Active research area - Linear Logical Framework by Cervesato and Pfenning 2002, works by Krishnaswami, Pradic, and Benton 2015 and Vákár 2017, based on Linear/Non-Linear logic Benton 1995

Linear logic and type theory

Adding linearity to dependent type theories:

- allows for precise usage tracking, viewing data as a computational resource.
- could enforce some restricted behaviour, e.g. in-place updates.
- could lead to more efficient implementations, e.g. erasability.

Active research area - Linear Logical Framework by Cervesato and Pfenning 2002, works by Krishnaswami, Pradic, and Benton 2015 and Vákár 2017, based on Linear/Non-Linear logic Benton 1995 , works by Orchard et al. 2019, Fu et al. 2020 and Abel and Bernardy 2020.

Linear logic and type theory

Adding linearity to dependent type theories:

- allows for precise usage tracking, viewing data as a computational resource.
- could enforce some restricted behaviour, e.g. in-place updates.
- could lead to more efficient implementations, e.g. erasability.

Active research area - Linear Logical Framework by Cervesato and Pfenning 2002, works by Krishnaswami, Pradic, and Benton 2015 and Vákár 2017, based on Linear/Non-Linear logic Benton 1995 , works by Orchard et al. 2019, Fu et al. 2020 and Abel and Bernardy 2020.

Quantitative Type Theory (QTT) (McBride 2016, Atkey 2018):

- offers a clear distinction between computation usage and type formation.

Linear logic and type theory

Adding linearity to dependent type theories:

- allows for precise usage tracking, viewing data as a computational resource.
- could enforce some restricted behaviour, e.g. in-place updates.
- could lead to more efficient implementations, e.g. erasability.

Active research area - Linear Logical Framework by Cervesato and Pfenning 2002, works by Krishnaswami, Pradic, and Benton 2015 and Vákár 2017, based on Linear/Non-Linear logic Benton 1995 , works by Orchard et al. 2019, Fu et al. 2020 and Abel and Bernardy 2020.

Quantitative Type Theory (QTT) (McBride 2016, Atkey 2018):

- offers a clear distinction between computation usage and type formation.
- always possible to contemplate already consumed things.

Linear logic and type theory

Adding linearity to dependent type theories:

- allows for precise usage tracking, viewing data as a computational resource.
- could enforce some restricted behaviour, e.g. in-place updates.
- could lead to more efficient implementations, e.g. erasability.

Active research area - Linear Logical Framework by Cervesato and Pfenning 2002, works by Krishnaswami, Pradic, and Benton 2015 and Vákár 2017, based on Linear/Non-Linear logic Benton 1995 , works by Orchard et al. 2019, Fu et al. 2020 and Abel and Bernardy 2020.

Quantitative Type Theory (QTT) (McBride 2016, Atkey 2018):

- offers a clear distinction between computation usage and type formation.
- always possible to contemplate already consumed things.
- type formation does not consume resources.

Anatomy of a QTT judgement

$$\mathsf{double} : \mathbb{N} \to \mathbb{N}, \mathsf{x} : \mathbb{N} \vdash \mathsf{double}\ \mathsf{x} : \mathbb{N}$$

Anatomy of a QTT judgement

$$\mathsf{double} \overset{1}{:} \mathbb{N} \to \mathbb{N}, \mathsf{x} \overset{2}{:} \mathbb{N} \vdash \mathsf{double}\ \mathsf{x} : \mathbb{N}$$

Anatomy of a QTT judgement

- annotations denote resources from an arbitrary usage semiring R

$$\mathsf{double} \overset{1}{:} \mathbb{N} \to \mathbb{N}, \mathsf{x} \overset{2}{:} \mathbb{N} \vdash \mathsf{double}\ \mathsf{x} : \mathbb{N}$$

Anatomy of a QTT judgement

- annotations denote resources from an arbitrary usage semiring R
  - typical choices for R are $\mathbb{N}$, $\{0, 1, \omega\}$

$$\mathsf{double} \overset{1}{:} \mathbb{N} \to \mathbb{N}, \mathsf{x} \overset{2}{:} \mathbb{N} \vdash \mathsf{double}\ \mathsf{x} : \mathbb{N}$$

Anatomy of a QTT judgement

- annotations denote resources from an arbitrary usage semiring R
  - typical choices for R are $\mathbb{N}$, $\{0, 1, \omega\}$

$$\mathsf{double} \overset{1}{:} \Pi\, z : \mathbb{N}.\, \mathbb{N}, \mathsf{x} \overset{2}{:} \mathbb{N} \vdash \mathsf{double}\ \mathsf{x} : \mathbb{N}$$

Anatomy of a QTT judgement

- annotations denote resources from an arbitrary usage semiring R
  - typical choices for R are $\mathbb{N}$, $\{0, 1, \omega\}$

$$\mathsf{double} \overset{1}{:} \Pi\, z \overset{2}{:} \mathbb{N}.\, \mathbb{N}, \mathsf{x} \overset{2}{:} \mathbb{N} \vdash \mathsf{double}\ \mathsf{x} : \mathbb{N}$$

Anatomy of a QTT judgement

- annotations denote resources from an arbitrary usage semiring R
  - typical choices for R are $\mathbb{N}$, $\{0, 1, \omega\}$

$$\mathsf{double} \overset{1}{:} \mathbb{N} \xrightarrow{2} \mathbb{N}, \mathsf{x} \overset{2}{:} \mathbb{N} \vdash \mathsf{double}\ \mathsf{x} : \mathbb{N}$$

Anatomy of a QTT judgement

- annotations denote resources from an arbitrary usage semiring R
  - typical choices for R are $\mathbb{N}$, $\{0, 1, \omega\}$

$$\mathsf{double} \overset{1}{:} \mathbb{N} \xrightarrow{2} \mathbb{N}, \mathsf{x} \overset{2}{:} \mathbb{N} \vdash \mathsf{double}\ \mathsf{x} \overset{1}{:} \mathbb{N}$$

Anatomy of a QTT judgement

- annotations denote resources from an arbitrary usage semiring R
  - typical choices for R are $\mathbb{N}$, $\{0, 1, \omega\}$

$$\text{double} \overset{1}{:} \mathbb{N} \xrightarrow{2} \mathbb{N}, \text{x} \overset{2}{:} \mathbb{N} \vdash \text{double x} \overset{1}{:} \mathbb{N}$$

- if the term double x on the rhs is annotated with σ, then:

Anatomy of a QTT judgement

- annotations denote resources from an arbitrary usage semiring R
  - typical choices for R are $\mathbb{N}$, $\{0, 1, \omega\}$

$$\mathsf{double} \overset{1}{:} \mathbb{N} \xrightarrow{2} \mathbb{N}, \mathsf{x} \overset{2}{:} \mathbb{N} \vdash \mathsf{double}\ \mathsf{x} \overset{1}{:} \mathbb{N}$$

- if the term double x on the rhs is annotated with σ, then:
  - $\sigma \in \{0, 1\}$

Anatomy of a QTT judgement

- annotations denote resources from an arbitrary usage semiring R
  - typical choices for R are $\mathbb{N}$, $\{0, 1, \omega\}$

$$\text{double} \overset{1}{:} \mathbb{N} \xrightarrow{2} \mathbb{N}, \text{x} \overset{2}{:} \mathbb{N} \vdash \text{double x} \overset{1}{:} \mathbb{N}$$

- if the term double x on the rhs is annotated with σ, then:
  - $\sigma \in \{0, 1\}$
  - 0 - double x is computationally irrelevant

Anatomy of a QTT judgement

- annotations denote resources from an arbitrary usage semiring R
  - typical choices for R are $\mathbb{N}$, $\{0, 1, \omega\}$

$$\mathsf{double} \overset{1}{:} \mathbb{N} \xrightarrow{2} \mathbb{N}, \mathsf{x} \overset{2}{:} \mathbb{N} \vdash \mathsf{double}\ \mathsf{x} \overset{1}{:} \mathbb{N}$$

- if the term double x on the rhs is annotated with σ, then:
  - $\sigma \in \{0, 1\}$
  - 0 - double x is computationally irrelevant
  - 1 - double x has computational content

Anatomy of a QTT judgement

- annotations denote resources from an arbitrary usage semiring R
  - typical choices for R are $\mathbb{N}$, $\{0, 1, \omega\}$

$$\mathsf{double} \overset{1}{:} \mathbb{N} \xrightarrow{2} \mathbb{N}, \mathsf{x} \overset{2}{:} \mathbb{N} \vdash \mathsf{double}\ \mathsf{x} \overset{1}{:} \mathbb{N}$$

- if the term double x on the rhs is annotated with σ, then:
  - $\sigma \in \{0, 1\}$
  - 0 - double x is computationally irrelevant
  - 1 - double x has computational content
- applying a predicate to double x

Anatomy of a QTT judgement

- annotations denote resources from an arbitrary usage semiring R
  - typical choices for R are $\mathbb{N}$, $\{0, 1, \omega\}$

$$\mathsf{double} \overset{1}{:} \mathbb{N} \overset{2}{\to} \mathbb{N}, \mathsf{x} \overset{2}{:} \mathbb{N} \vdash \mathsf{double}\ \mathsf{x} \overset{1}{:} \mathbb{N}$$

- if the term double x on the rhs is annotated with σ, then:
  - $\sigma \in \{0, 1\}$
  - 0 - double x is computationally irrelevant
  - 1 - double x has computational content
- applying a predicate to double x

$$\mathsf{n} \overset{0}{:} \mathbb{N} \vdash \mathsf{even(n)} \overset{0}{:} \mathsf{Type}$$

Anatomy of a QTT judgement

- annotations denote resources from an arbitrary usage semiring R
  - typical choices for R are $\mathbb{N}$, $\{0, 1, \omega\}$

$$\mathsf{double} \overset{1}{:} \mathbb{N} \xrightarrow{2} \mathbb{N}, \mathsf{x} \overset{2}{:} \mathbb{N} \vdash \mathsf{double\ x} \overset{1}{:} \mathbb{N}$$

- if the term double x on the rhs is annotated with σ, then:
  - $\sigma \in \{0, 1\}$
  - 0 - double x is computationally irrelevant
  - 1 - double x has computational content
- applying a predicate to double x

$$\mathsf{n} \overset{0}{:} \mathbb{N} \vdash \mathsf{even(n)} \overset{0}{:} \mathsf{Type}$$

$$\mathsf{double} \overset{0}{:} \mathbb{N} \xrightarrow{2} \mathbb{N}, \mathsf{x} \overset{0}{:} \mathbb{N} \vdash \mathsf{even(double\ x)} \overset{0}{:} \mathsf{Type}$$

Quantitative containers

A (extension of) container in ordinary type theory is given by a functor:

$$F(X) := \Sigma(s : S).\, P(s) \to X$$

Quantitative containers

A (extension of) container in ordinary type theory is given by a functor:

$$F(X) := \Sigma(s : S).\, P(s) \to X$$

A direct translation in QTT yields:

$$F(X) := (s \overset{1}{:} S) \otimes (P(s) \xrightarrow{1} X)$$

Quantitative containers

A (extension of) container in ordinary type theory is given by a functor:

$$F(X) := \Sigma(s : S).\, P(s) \to X$$

A direct translation in QTT yields:

$$F(X) := (s \overset{1}{:} S) \otimes (P(s) \xrightarrow{1} X)$$

Proposition

Let $\mathcal{C}$ be the category of closed types and linear functions:

- objects - types $\vdash X$
- morphisms - functions $\vdash f : X \xrightarrow{1} Y$

The mapping $F_{S,P}(X) = (S \overset{1}{:} S) \otimes (P(s) \xrightarrow{1} X)$ is a functor on $\mathcal{C}$ for fixed S : Type and $P : S \xrightarrow{0}$ Type.

Quantitative containers

A (extension of) container in ordinary type theory is given by a functor:

$$F(X) := \Sigma(s : S).\, P(s) \to X$$

A direct translation in QTT yields:

$$F(X) := (s \overset{1}{:} S) \otimes (P(s) \xrightarrow{1} X)$$

Proposition

Let $\mathcal{C}$ be the category of closed types and linear functions:

- objects - types $\vdash X$
- morphisms - functions $\vdash f : X \xrightarrow{1} Y$

The mapping $F_{S,P}(X) = (S \overset{1}{:} S) \otimes (P(s) \xrightarrow{1} X)$ is a functor on $\mathcal{C}$ for fixed S : Type and $P : S \xrightarrow{0}$ Type.

We call any functor isomorphic to one of the form $F_{S,P}$ a quantitative container.

Induction principle

Let $\mathbf{W} := (W, c : F_{S,P}(W) \xrightarrow{1} W)$ be an $F_{S,P}$-algebra.

Induction principle

Let $\mathbf{W} := (W, c : F_{S,P}(W) \xrightarrow{1} W)$ be an $F_{S,P}$-algebra.

Induction principle

$$w \overset{0}{:} W \vdash Q(w) \overset{0}{:} \text{Type}$$

Induction principle

Let $\mathbf{W} := (W, c : F_{S,P}(W) \xrightarrow{1} W)$ be an $F_{S,P}$-algebra.

Induction principle

$$\frac{w \overset{0}{:} W \vdash Q(w) \overset{0}{:} \mathrm{Type}}{\vdash \mathrm{elim}(Q, M) \overset{1}{:} (w \overset{1}{:} W) \to Q(w)}$$

Let $\mathbf{W} := (W, c : F_{S,P}(W) \xrightarrow{1} W)$ be an $F_{S,P}$-algebra.

Induction principle

$$\frac{w \overset{0}{:} W \vdash Q(w) \overset{0}{:} \mathrm{Type} \qquad \vdash M \overset{1}{:} (s \overset{1}{:} S) \to}{\vdash \mathrm{elim}(Q, M) \overset{1}{:} (w \overset{1}{:} W) \to Q(w)}$$

the shape

Induction principle

Let $\mathbf{W} := (W, c : F_{S,P}(W) \xrightarrow{1} W)$ be an $F_{S,P}$-algebra.

Induction principle

$$\frac{\begin{array}{cr} w \overset{0}{:} W \vdash Q(w) \overset{0}{:} \mathrm{Type} & \\ \vdash M \overset{1}{:} (s \overset{1}{:} S) \to & \textit{the shape} \\ (h \overset{0}{:} P(s) \xrightarrow{1} W) \to & \textit{the positions} \end{array}}{\vdash \mathrm{elim}(Q, M) \overset{1}{:} (w \overset{1}{:} W) \to Q(w)}$$

Induction principle

Let $\mathbf{W} := (W, c : F_{S,P}(W) \xrightarrow{1} W)$ be an $F_{S,P}$-algebra.

Induction principle

$$\frac{\begin{array}{lr} w \overset{0}{:} W \vdash Q(w) \overset{0}{:} \mathrm{Type} & \\ \vdash M \overset{1}{:} (s \overset{1}{:} S) \to & \textit{the shape} \\ (h \overset{0}{:} P(s) \xrightarrow{1} W) \to & \textit{the positions} \\ \color{red}{((p \overset{1}{:} P(s)) \to Q(h(p)))} \xrightarrow{1} & \textit{i.h.} \end{array}}{\vdash \mathrm{elim}(Q, M) \overset{1}{:} (w \overset{1}{:} W) \to Q(w)}$$

Induction principle

Let $\mathbf{W} := (W, c : F_{S,P}(W) \xrightarrow{1} W)$ be an $F_{S,P}$-algebra.

Induction principle

$$\frac{\begin{array}{lr} w \overset{0}{:} W \vdash Q(w) \overset{0}{:} \mathrm{Type} & \\ \vdash M \overset{1}{:} (s \overset{1}{:} S) \to & \textit{the shape} \\ (h \overset{0}{:} P(s) \xrightarrow{1} W) \to & \textit{the positions} \\ ((p \overset{1}{:} P(s)) \to Q(h(p))) \xrightarrow{1} & \textit{i.h.} \\ Q(c(s,h)) & \end{array}}{\vdash \mathrm{elim}(Q, M) \overset{1}{:} (w \overset{1}{:} W) \to Q(w)}$$

Induction via initiality

Theorem

If $\mathbf{W}$ *is initial, the induction principle holds.*

Induction via initiality

Theorem

If **W** *is initial, the induction principle holds.*

Proof. Adapted from Hermida and Jacobs 1998, Awodey, Gambino, and Sojakova 2017.

- construct an $F_{S,P}$-algebra for $(w \overset{0}{:} W) \otimes Q$.

$$\begin{tikzcd}
F_{S,P}(W) \arrow[r, "c"] \arrow[d] & W \arrow[d, "\mathrm{fold}"] \\
F_{S,P}((w \overset{0}{:} W) \otimes Q) \arrow[r] \arrow[d] & (w \overset{0}{:} W) \otimes Q \arrow[d, "\mathrm{fst}"] \\
F_{S,P}(W) \arrow[r, "c"] & W
\end{tikzcd}$$

Induction via initiality

Theorem

If **W** *is initial, the induction principle holds.*

Proof. Adapted from Hermida and Jacobs 1998, Awodey, Gambino, and Sojakova 2017.

$$
\begin{tikzcd}
F_{S,P}(W) \arrow[r, "c"] \arrow[d] & W \arrow[d, "\mathsf{fold}"] \\
F_{S,P}((w \overset{0}{:} W) \otimes Q) \arrow[r] \arrow[d] & (w \overset{0}{:} W) \otimes Q \arrow[d, "\mathsf{fst}"] \\
F_{S,P}(W) \arrow[r, "c"] & W
\end{tikzcd}
$$

- construct an $F_{S,P}$-algebra for $(w \overset{0}{:} W) \otimes Q$.
- compose the mediating morphism $\mathsf{fold} : W \xrightarrow{1} (w \overset{0}{:} W) \otimes Q$ with $\mathsf{snd} : (x \overset{1}{:} (w \overset{0}{:} W) \otimes Q) \to Q(\mathsf{fst}(x))$.

Induction via initiality

Theorem

If $\mathbf{W}$ *is initial, the induction principle holds.*

Proof. Adapted from Hermida and Jacobs 1998, Awodey, Gambino, and Sojakova 2017.

$$\begin{tikzcd}
F_{S,P}(W) \arrow[r, "c"] \arrow[d] & W \arrow[d, "\mathsf{fold}"] \\
F_{S,P}((w \overset{0}{:} W) \otimes Q) \arrow[r] \arrow[d] & (w \overset{0}{:} W) \otimes Q \arrow[d, "\mathsf{fst}"] \\
F_{S,P}(W) \arrow[r, "c"] & W
\end{tikzcd}$$

- construct an $F_{S,P}$-algebra for $(w \overset{0}{:} W) \otimes Q$.
- compose the mediating morphism $\mathsf{fold} : W \xrightarrow{1} (w \overset{0}{:} W) \otimes Q$ with $\mathsf{snd} : (x \overset{1}{:} (w \overset{0}{:} W) \otimes Q) \to Q(\mathsf{fst}(x))$.
- show that the map fst is an $F_{S,P}$-algebra morphism and so is the composite $\mathsf{fst} \circ \mathsf{fold} : W \xrightarrow{1} W$.

Induction via initiality

Theorem

If **W** *is initial, the induction principle holds.*

Proof. Adapted from Hermida and Jacobs 1998, Awodey, Gambino, and Sojakova 2017.

$$
\begin{tikzcd}
F_{S,P}(W) \arrow[r, "c"] \arrow[d] & W \arrow[d, "\mathsf{fold}"] \\
F_{S,P}((w \overset{0}{:} W) \otimes Q) \arrow[r] \arrow[d] & (w \overset{0}{:} W) \otimes Q \arrow[d, "\mathsf{fst}"] \\
F_{S,P}(W) \arrow[r, "c"] & W
\end{tikzcd}
$$

- construct an $F_{S,P}$-algebra for $(w \overset{0}{:} W) \otimes Q$.
- compose the mediating morphism $\mathsf{fold} : W \overset{1}{\to} (w \overset{0}{:} W) \otimes Q$ with $\mathsf{snd} : (x \overset{1}{:} (w \overset{0}{:} W) \otimes Q) \to Q(\mathsf{fst}(x))$.
- show that the map fst is an $F_{S,P}$-algebra morphism and so is the composite $\mathsf{fst} \circ \mathsf{fold} : W \overset{1}{\to} W$.
- it follows that $\mathsf{fst} \circ \mathsf{fold} = \mathsf{id}$.

Strictly positive types

A polynomial functor $F(X)$ traditionally can also be presented as a strictly positive type.

Definition (Strictly positive type)

A *(non-inductive)* strictly positive type over a type variable X is type expression generated by:

$$X \mid K \mid F \times G \mid F + G \mid K \to F$$

where F and G are strictly positive types and K is a closed type (with no type variables).

Strictly positive types

A polynomial functor $F(X)$ traditionally can also be presented as a strictly positive type.

Definition (Strictly positive type)

A *(non-inductive)* strictly positive type over a type variable X is type expression generated by:

$$X \mid K \mid F \times G \mid F + G \mid K \to F$$

where F and G are strictly positive types and K is a closed type (with no type variables).

Theorem (Abbott, Altenkirch, and Ghani 2005)

Every non-inductive strictly positive type can be represented as a container.

Quantitative polynomial functors I

However, container representation breaks down in QTT:

Quantitative polynomial functors I

However, container representation breaks down in QTT:

$$\mathbf{0} \xrightarrow{1} X \cong \mathbf{T} \ncong \mathbf{I}$$

$$\mathbf{2} \xrightarrow{1} X \cong X \,\&\, X \ncong X \otimes X$$

Quantitative polynomial functors I

However, container representation breaks down in QTT:

$$\mathbf{0} \xrightarrow{1} X \cong \mathbf{T} \ncong \mathbf{I}$$

$$\mathbf{2} \xrightarrow{1} X \cong X \mathbin{\&} X \ncong X \otimes X$$

But we can still inductively generate the class of quantitative polynomial functors (QPF) by:

$$\mathsf{Id} \mid \mathsf{Const}_K \mid F \otimes G \mid F \oplus G \mid \; F \mathbin{\&} G \mid K \to -$$

where F and G are QPFs and K - a closed type.

Quantitative polynomial functors I

However, container representation breaks down in QTT:

$$\mathbf{0} \xrightarrow{1} X \cong \mathbf{T} \ncong \mathbf{I}$$

$$\mathbf{2} \xrightarrow{1} X \cong X \,\&\, X \ncong X \otimes X$$

But we can still inductively generate the class of quantitative polynomial functors (QPF) by:

$$\mathsf{Id} \mid \mathsf{Const}_K \mid F \otimes G \mid F \oplus G \mid \; F \,\&\, G \mid K \to -$$

where F and G are QPFs and K - a closed type.
We can recover the induction principle for QPFs as well.

Quantitative polynomial functors II

Manually define a predicate lifting $\widehat{F}_X : (Q : X \to \text{Type}) \to (F(X) \to \text{Type})$ to encode the induction hypothesis:

Quantitative polynomial functors II

Manually define a predicate lifting $\widehat{F}_X : (Q : X \to \mathrm{Type}) \to (F(X) \to \mathrm{Type})$ to encode the induction hypothesis:

$$\widehat{\mathrm{Id}}(Q,z) := Q(z) \qquad \widehat{F \otimes G}(Q,z) := \widehat{F}(Q, \mathrm{fst}\ z) \otimes \widehat{G}(Q, \mathrm{snd}\ z)$$

$$\widehat{\mathrm{Const}_K}(Q,z) := K \qquad \ldots$$

Quantitative polynomial functors II

Manually define a predicate lifting $\widehat{F}_X : (Q : X \to \text{Type}) \to (F(X) \to \text{Type})$ to encode the induction hypothesis:

$$\widehat{\text{Id}}(Q, z) := Q(z) \qquad \widehat{F \otimes G}(Q, z) := \widehat{F}(Q, \text{fst } z) \otimes \widehat{G}(Q, \text{snd } z)$$

$$\widehat{\text{Const}_K}(Q, z) := K \qquad \ldots$$

Let F be a QPF and $\mathbf{W} := (W, c : F(W) \xrightarrow{1} W)$ - an F-algebra.

Induction principle for QPFs

$$\frac{w \overset{0}{:} W \vdash Q(w) \overset{0}{:} \text{Type} \qquad \vdash M \overset{1}{:} ((w \overset{0}{:} F(W)) \otimes \widehat{F}(Q, w)) \xrightarrow{1} \xrightarrow{1} Q(c(w))}{\vdash \text{elim}(Q, M) : (w \overset{1}{:} W) \to Q(w)}$$

Theorem

The induction principle holds if $\mathbf{W}$ *is initial.*

Proof. Use a distributive lemma for the dependent tensor and the predicate lifting:

Lemma

$$F((w \overset{0}{:} W) \otimes Q) \cong (w \overset{0}{:} F(W)) \otimes \widehat{F}(Q, w).$$

Summary and further directions

What we have done so far:

Summary and further directions

What we have done so far:

- given a description of a quantitative container and a quantitative polynomial functor

Summary and further directions

What we have done so far:

- given a description of a quantitative container and a quantitative polynomial functor
- derived an induction principle for QPFs assuming initiality

Summary and further directions

What we have done so far:

- given a description of a quantitative container and a quantitative polynomial functor
- derived an induction principle for QPFs assuming initiality
- shown the existence of initial algebras for finitary QPFs in a realisability model

Summary and further directions

What we have done so far:

- given a description of a quantitative container and a quantitative polynomial functor
- derived an induction principle for QPFs assuming initiality
- shown the existence of initial algebras for finitary QPFs in a realisability model

We hope to extend this work by:

- giving a semantic characterisation of QPFs.

Summary and further directions

What we have done so far:

- given a description of a quantitative container and a quantitative polynomial functor
- derived an induction principle for QPFs assuming initiality
- shown the existence of initial algebras for finitary QPFs in a realisability model

We hope to extend this work by:

- giving a semantic characterisation of QPFs.
- constructing initial algebras for non-finitary QPFs.

Thank you for your attention!

References I

Michael Abbott, Thorsten Altenkirch, and Neil Ghani. "Categories of containers". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 2620 (2003). ISBN: 3540008977, pp. 23–38. (Visited on 04/14/2021).

Michael Abbott, Thorsten Altenkirch, and Neil Ghani. "Containers: Constructing strictly positive types". en. In: *Theoretical Computer Science*. Applied Semantics: Selected Topics 342.1 (Sept. 2005), pp. 3–27. URL: https://www.sciencedirect.com/science/article/pii/S0304397505003373 (visited on 08/28/2021).

Steve Awodey, Nicola Gambino, and Kristina Sojakova. "Homotopy-Initial Algebras in Type Theory". In: *Journal of the ACM* 63.6 (2017).

References II

Robert Atkey. "Syntax and Semantics of Quantitative Type Theory". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science - LICS '18*. ISSN: 10436871. New York, New York, USA: ACM Press, 2018, pp. 56–65. URL: http://dl.acm.org/citation.cfm?doid=3209108.3209189.

Nick Benton. "A mixed linear and non-linear logic: Proofs, terms and models". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 933. ISSN: 16113349. Springer, Berlin, Heidelberg, 1995, pp. 121–135. URL: http://link.springer.com/10.1007/BFb0022251.

Edwin Brady. "Idris 2: Quantitative Type Theory in Practice". In: *arXiv:2104.00480 [cs]* (Apr. 2021). arXiv: 2104.00480. URL: http://arxiv.org/abs/2104.00480 (visited on 08/31/2021).

References III

Iliano Cervesato and Frank Pfenning. “A Linear Logical Framework”. In: *Information and Computation* 179.1 (Nov. 2002). Publisher: Academic Press, pp. 19–75. URL: https://linkinghub.elsevier.com/retrieve/pii/S0890540101929517 (visited on 04/23/2021).

Claudio Hermida and Bart Jacobs. “Structural Induction and Coinduction in a Fibrational Setting”. In: *Information and Computation* 145.2 (Sept. 1998). Publisher: Academic Press, pp. 107–152. URL: https://linkinghub.elsevier.com/retrieve/pii/S0890540198927250 (visited on 03/18/2021).

References IV

Neelakantan R. Krishnaswami, Pierre Pradic, and Nick Benton. "Integrating Linear and Dependent Types". In: *ACM SIGPLAN Notices* 50.1 (May 2015). Publisher: Association for Computing Machinery (ACM), pp. 17–30. URL: https://dl.acm.org/doi/10.1145/2775051.2676969 (visited on 04/23/2021).

Conor McBride. "I Got Plenty o' Nuttin'". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 9600. ISSN: 16113349. Springer Verlag, 2016, pp. 207–233. URL: http://link.springer.com/10.1007/978-3-319-30936-1_12 (visited on 09/27/2020).

References V

Dominic Orchard, Vilem-Benjamin Liepelt, and Harley Eades III. "Quantitative program reasoning with graded modal types". In: *Proceedings of the ACM on Programming Languages* 3.ICFP (July 2019), 110:1–110:30. URL: https://doi.org/10.1145/3341714 (visited on 06/11/2021).

Matthijs Vákár. "In search of effectful dependent types". PhD thesis. University of Oxford, 2017.