Adventures in Quantitative Type Theory

Stephanie Weirich
University of Pennsylvania
Joint work with Pritam Choudhury

WG 2.8 Zion - 2020

A month in the laboratory can often save an hour in the library.

--Frank Westheimer

An hour in the library can lead to a month in the laboratory.

Recent work

- Conor McBride. *I Got Plenty o’ Nuttin’.* WadlerFest 2016
- Bernardy et al. *Linear Haskell: practical linearity in a higher-order polymorphic language*. POPL 2018
- Bob Atkey. *The Syntax and Semantics of Quantitative Type Theory*. LICS 2018
- Andreas Abel. *On the Syntax and Semantics of Quantitative Typing.* Talk given at Workshop on Mixed Inductive-Coinductive Reasoning. April 2018
- Orchard, Liepelt, Eades. *Quantitative Program Reasoning with Graded Modal Types*. ICFP 2019
- And many more…

Motivation: Haskell + Dependent Types

- Dependent Haskell should be able to express *run-time* irrelevance (i.e. it should preserve Haskell's type erasure semantics)
- Current status: ICC-based. Curry-style system that omits irrelevant arguments from terms.

  $\Gamma \vdash a : \forall (x{:}Type),\ x \to x$

  $\Gamma \vdash a\ \Box\ 3 : \mathrm{Int} \to \mathrm{Int}$
- Current oops: I don't see how to make this work for Strong-Σ types

  $\Gamma \vdash (\Box,\ 3) : \exists (x{:}Type).\ x$

  $\Gamma,\ y{:}\ (\exists x{:}Type.x) \vdash \mathrm{snd}\ \ y : \mathrm{fst}\ \ y$

Enter Quantitative Type Theory

- Want to replace mechanism for irrelevance.
- Inspiration: LinearHaskell is based on Quantitative Type Theory.
- McBride/Atkey showed that a dependently-typed version of Quantitative Type Theory can generalize irrelevance (things used 0-times) and linearity (things used 1 time).

Let's use that.

What is QTT?

- Tracks the demands that computations make on the context
- Generalizes linear types (cf. Linear Haskell) and Bounded Linear Types
- Also useful for
  - Irrelevance in dependent type theory (cf. Agda, Idris)
  - Security levels
  - Differential privacy
  - Cardinality analysis in compilers
- Related to coeffects and graded comonads (cf. Granule project)

Quantitative Type Theory

- Typing context tracks the number of times a variable is *used*
- **Flexible**: Quantities (aka resources or usages) come described by an arbitrary semiring
- Operations must satisfy usual properties of semirings (i.e. identity elements, assoc, comm, distributivity)

$$\Gamma ::= \varnothing \mid \Gamma, x :^{q} A$$

$$\begin{array}{lcl} q & ::= & 0 \mid 1 \mid \ldots \\ & \mid & q_1 + q_2 \\ & \mid & q_1 \cdot q_2 \end{array}$$

QTT: additional operations

- Scalar multiplication $q \cdot \Gamma$
- Pointwise addition $\Gamma_1 + \Gamma_2$
- Resource partial order $q_1 \leq q_2$
  - $q_1 \leq q_2$ implies $q_1 + r \leq q_2 + r$
  - $q_1 \leq q_2$ implies $q_1 \cdot r \leq q_2 \cdot r$

Contexts must exactly match i.e. this operation is only defined when $0 \cdot \Gamma_1 = 0 \cdot \Gamma_2$

Quantitative type system (Version 1)

$$\frac{}{0 \cdot \Gamma_1, x :^1 A, 0 \cdot \Gamma_2 \vdash x : A}$$

Record exactly one variable use

$$\frac{\Gamma, x :^q A \vdash a : B}{\Gamma \vdash \lambda x . a : (A \to^q B)}$$

Record how many x's are needed in the function type

$$\frac{\Gamma_1 \vdash a : (A \to^q B) \qquad \Gamma_2 \vdash b : A}{\Gamma_1 + (q \cdot \Gamma_2) \vdash a\, b : B}$$

Multiply the resources needed by the argument in an application

$$\frac{\Gamma_1 \vdash a : A \qquad \Gamma_1 \leq \Gamma_2}{\Gamma_2 \vdash a : A}$$

Can ask for *more* than you need (optional rule)

Simple Example: Boolean Semiring

- Semiring includes *only* 0 (absent), 1 = ω (present)
- NOT linear types, because 1 + 1 = 1
- Sample derivation

$$\Gamma_1 = x :^0 \mathbf{Int}, f :^1 (\mathbf{Int} \to^0 \mathbf{Int})$$
$$\Gamma_2 = x :^1 \mathbf{Int}, f :^0 (\mathbf{Int} \to^0 \mathbf{Int})$$

$$\frac{\Gamma_1 \vdash f : \mathbf{Int} \to^0 \mathbf{Int} \qquad \Gamma_2 \vdash x : \mathbf{Int}}{\Gamma_1 + 0 \cdot \Gamma_2 \vdash f\ x : \mathbf{Int}}$$

Simple Example: Subusage

- Semiring includes *only* 0 (absent), 1 = ω (present)
- More typing derivations available with $0 \leq \omega$
- Degenerates to STLC

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x :^1 \mathbf{Int}, y :^0 \mathbf{Int} \vdash x : \mathbf{Int}}}{x :^1 \mathbf{Int}, y :^1 \mathbf{Int} \vdash x : \mathbf{Int}}}{x :^1 \mathbf{Int} \vdash \lambda y.x : \mathbf{Int} \to^1 \mathbf{Int}}}{\varnothing \vdash \lambda x.\lambda y.x : \mathbf{Int} \to^1 \mathbf{Int} \to^1 \mathbf{Int}} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{x :^0 \mathbf{Int}, y :^1 \mathbf{Int} \vdash y : \mathbf{Int}}}{x :^1 \mathbf{Int}, y :^1 \mathbf{Int} \vdash y : \mathbf{Int}}}{x :^1 \mathbf{Int} \vdash \lambda y.y : \mathbf{Int} \to^1 \mathbf{Int}}}{\varnothing \vdash \lambda x.\lambda y.y : \mathbf{Int} \to^1 \mathbf{Int} \to^1 \mathbf{Int}}$$

Linearity

- Semiring includes 0 (absent), 1 (linear), ω (unrestricted)
- Only include reflexivity in ≤
- 1 + 1 = ω, so variables marked 1 may be used only once

- [Exercise for audience: work out another example]

Example: Linearity++

$$
\begin{aligned}
\mathbf{irr} &\triangleq \{0\} \\
\mathbf{lin} &\triangleq \{1\} \\
\mathbf{aff} &\triangleq \{0, 1\} \\
\mathbf{rel} &\triangleq \{1, 2, \ldots\} \\
\mathbf{unr} &\triangleq \{0, 1, 2, \ldots\}
\end{aligned}
$$

- Five elements in semiring
- Have irr = 0 and lin = 1
- Have to approximate (rel + rel = rel)
- Do not have (or want) 0 ≤ 1
- Valid to omit aff & rel
- Valid to include all subsets of ℕ (Bounded Linear Logic)

$$
\begin{aligned}
q_1 + q_2 &\triangleq \{n_1 + n_2 \mid n_1 \in q_1, n_2 \in q_2\} \\
q_1 \cdot q_2 &\triangleq \{n_1 \cdot n_2 \mid n_1 \in q_1, n_2 \in q_2\} \\
q_1 \leq q_2 &\triangleq q_1 \subseteq q_2
\end{aligned}
$$

Properties

Lemma 1 (Substitution) *If* $\Gamma \vdash a : A$ *and* $\Gamma_1, x :^q A, \Gamma_2 \vdash b : B$ *then* $(\Gamma_1 + q \cdot \Gamma), \Gamma_2 \vdash b\{a/x\} : B$

Lemma 2 (Weakening) *If* $\Gamma_1, \Gamma_2 \vdash b : B$ *then* $\Gamma_1, x :^0 A, \Gamma_2 \vdash b : B$

Lemma 3 (Preservation) *If* $\Gamma \vdash a : A$ *and* $\vdash a \rightsquigarrow a'$ *then* $\Gamma \vdash a' : A$.

What about dependent types?

- Want to separate runtime uses of a variable (resourced) from compile-time uses (unrestricted) so that variables can appear freely in types
- Conor's idea: Distinguish by annotating the judgement

  $\Gamma \vdash a :^1 A$ corresponds to $\Gamma \vdash a : A$

  $\Gamma \vdash a :^0 A$ only uses compile-time resources
- General multiplication principle

  If $\Gamma \vdash a :^q A$ then $r{\cdot}\Gamma \vdash a :^{r \cdot q} A$

Quantitative type system (Version 2)

$$\frac{}{0\cdot\Gamma_1, x :^1 A, 0\cdot\Gamma_2 \vdash x : A} \qquad \frac{\Gamma, x :^q A \vdash a : B}{\Gamma \vdash \lambda x.a : (A \to^q B)}$$

$$\frac{}{0\cdot\Gamma_1, x :^q A, 0\cdot\Gamma_2 \vdash x :^q A} \qquad \frac{\Gamma, x :^{q_0 \cdot q} A \vdash a :^{q_0} B}{\Gamma \vdash \lambda x.a :^{q_0} (A \to^q B)}$$

Quantitative type system (Version 2)

$$\frac{\begin{array}{c}\Gamma_1 \vdash a : A \\ \Gamma_1 \leq \Gamma_2\end{array}}{\Gamma_2 \vdash a : A} \qquad \frac{\begin{array}{c}\Gamma_1 \vdash a : (A \to^q B) \\ \Gamma_2 \vdash b : A\end{array}}{\Gamma_1 + (q \cdot \Gamma_2) \vdash a\, b : B}$$

$$\frac{\begin{array}{c}\Gamma_1 \vdash a :^q A \\ \Gamma_1 \leq \Gamma_2\end{array}}{\Gamma_2 \vdash a :^q A} \qquad \frac{\begin{array}{c}\Gamma_1 \vdash a :^{q_0} (A \to^q B) \\ \Gamma_2 \vdash b :^{q_0} A\end{array}}{\Gamma_1 + (q \cdot \Gamma_2) \vdash a\, b :^{q_0} B}$$

What is wrong with this rule?

$$\frac{\Gamma_1 \vdash a :^{q_0} (A \to^{q} B) \qquad \Gamma_2 \vdash b :^{q_0} A}{\Gamma_1 + (q \cdot \Gamma_2) \vdash a\ b :^{q_0} B}$$

- Preservation theorem requires a substitution lemma
- The substitution lemma needs the following two properties, which do not hold for arbitrary semirings.

Lemma 4 (Splitting) *If* $\Gamma \vdash a :^{q+r} A$ *then there exists* $\Gamma_1 + \Gamma_2 = \Gamma$ *such that* $\Gamma_1 \vdash a :^{q} A$ *and* $\Gamma_1 \vdash a :^{r} A$.

Lemma 5 (Factoring) *If* $\Gamma \vdash a :^{q \cdot r} A$ *then there exists* $q \cdot \Gamma_1 = \Gamma$ *such that* $\Gamma_1 \vdash a :^{r} A$.

What can we do instead?

$$\frac{\Gamma_1 \vdash a :^{\sigma_0} (A \to^q B) \quad \Gamma_2 \vdash b :^{\sigma_1} A \quad \sigma_1 = 0 \textbf{ iff } (\sigma_0 \cdot q = 0)}{\Gamma_1 + q \cdot \Gamma_2 \vdash a\, b :^{\sigma_0} B}$$

- Atkey: restrict usages to those that we *can* factor. i.e. only 0 or 1

- Us: revise rules with special cases for 0 and 1 (no need to restrict judgement overall)

App

$$\frac{\Gamma_1 \vdash a :^{q_0} (A \to^q B) \quad \Gamma_2 \vdash b :^{1} A}{\Gamma_1 + (q \cdot q_0) \cdot \Gamma_2 \vdash a\, b :^{q_0} B}$$

App0

$$\frac{\Gamma_1 \vdash a :^{q_0} (A \to^q B) \quad q_0 \cdot q = 0 \quad \Gamma_2 \vdash b :^{0} A}{\Gamma_1 \vdash a\, b :^{q_0} B}$$

Two different substitution lemmas

Lemma 6 (Substitution) *If* $\Gamma \vdash a :^{q_1} A$ *and* $\Gamma_1, x :^{q_1} A, \Gamma_2 \vdash b :^{q_2} B$ *then* $(\Gamma_1 + q_1 \cdot \Gamma), \Gamma_2 \vdash b\{a/x\} :^{q_2} B$

Lemma 7 (Substitution-1) *If* $\Gamma \vdash a :^{1} A$ *and* $\Gamma_1, x :^{q_1} A, \Gamma_2 \vdash b :^{q_2} B$ *then* $(\Gamma_1 + q_1 \cdot \Gamma), \Gamma_2 \vdash b\{a/x\} :^{q_2} B$

Lemma 8 (Substitution-0) *If* $\Gamma \vdash a :^{0} A$ *and* $\Gamma_1, x :^{0} A, \Gamma_2 \vdash b :^{q} B$ *then* $\Gamma_1, \Gamma_2 \vdash b\{a/x\} :^{q} B$

Irrelevance needs more from semiring

Lemma 8 (Substitution-0) *If* $\Gamma \vdash a :^0 A$ *and* $\Gamma_1, x :^0 A, \Gamma_2 \vdash b :^q B$ *then* $\Gamma_1, \Gamma_2 \vdash b\{a/x\} :^q B$

- For all q, $q \leq 0$ implies $q = 0$.
- For all q, r, $q + r = 0$ implies $q = 0$ and $r = 0$.
- For all q, r, $q \cdot r = 0$ implies $q = 0$ or $r = 0$.

Where are we?

- New variant of Atkey's fix of Conor's version of QTT
  - Slightly modified application rule
  - Addition of subusage rule
- CAVEAT: So far, simple types only
- Weakening, two Substitution lemmas, and Preservation theorem proved in Coq

[NOTE: Coq is not really the right tool for this work.]

What next?

- Extend to dependent types
- Also, exploring alternative approaches that do *not* add usage annotations to the judgement.

- Longer term: subtyping? usage polymorphism?
- Longer term: relation to graded modal types?