Resource Constrained Programming with Full Dependent Types

Robert Atkey
Strathclyde University, Glasgow
robert.atkey@strath.ac.uk

IRIF, 23rd January 2020

Goal of this talk —

- To extend Implicit Computational Complexity to Dependent Types
- Motivations:
  - Certified resource constrained programming
  - Resource constrained mathematics?
- Using *Quantitative Type Theory*.

Programming in Type Theory

Programming in Type Theory —

$$\vdash M : \mathrm{Vec\ Nat\ 5}$$

$$\vdash \mathrm{check} : (t : \mathrm{Tm}) \longrightarrow (a : \mathrm{Ty}) \longrightarrow (\vdash'\ t\ a) + \neg(\vdash'\ t\ a)$$

- An expressive typed functional language
- Implemented in Agda, Coq, Idris, Lean, ...

Programming in Type Theory —

$$
\begin{array}{l}
\textbf{data}\ \mathit{Vec}\ (A : \mathrm{Set}) : \mathrm{Nat} \longrightarrow \mathrm{Set}\ \textbf{where} \\
\quad \mathrm{nil} : \mathrm{Vec}\ A\ 0 \\
\quad \mathrm{cons} : (n : \mathrm{Nat}) \longrightarrow A \longrightarrow \mathrm{Vec}\ A\ n \longrightarrow \mathrm{Vec}\ A\ (\mathrm{S}\ n)
\end{array}
$$

- Standard compilation scheme —
  - cons constructor has three arguments
  - Space consumption is $O(n^2)$!

Type Theory is tough to compile —

- Extraneous data must be erased

  e.g., sizes in vectors

- The compilation target is assumed to have unlimited capacity —
  - In time and space

Type Theory is tough to compile —

- Extraneous data must be erased

  e.g., sizes in vectors

- The compilation target is assumed to have unlimited capacity —
  - In **time** and space

Constrain the resources required —

- Use *Quantitative Type Theory* to control resource usage
- Use Implicit Computational Complexity ideas to restrict time complexity

Two Systems by Martin Hofmann

Implicit Computational Complexity —

- Capturing complexity classes via restricted functional languages
- Closely related to static analysis for resource consumption
- Going to look at two systems by Martin Hofmann:
  - LFPL "*Linear Function Programming Language*"
  - Amortised Resource Analysis
- Controlling (nested) iteration is the key.

Warm-up: Linear time system —

- In the linear λ-calculus
- A natural number iterator:

$$\frac{\Gamma_1 \vdash M_z : A \qquad x : A \vdash M_s : A \qquad \Gamma_2 \vdash N : \mathrm{Nat}}{\Gamma_1, \Gamma_2 \vdash \mathrm{iter}_A(M_z, x.M_s, N) : A}$$

- Once we iterate over some number N it is *used up*

Warm-up: Linear time system —

- In the linear λ-calculus
- A natural number iterator:

$$\frac{\Gamma_1 \vdash M_z : A \qquad x : A \vdash M_s : A \qquad \Gamma_2 \vdash N : \mathrm{Nat}}{\Gamma_1, \Gamma_2 \vdash \mathrm{iter}_A(M_z, x.M_s, N) : A}$$

- Once we iterate over some number N it is *used up*
- No way to construct new “iterable” natural numbers
  - If we could, then could easily duplicate Nats
  - then iterate to get multiplication
  - and again to get exponentiation
- Can have other constructable data that is not iterable: Nat°

LFPL: extending to polynomial time

- Introduce a type $\diamond$, representing a chunk of iterability
- Building a Nat requires $\diamond$s:

$$0 : \diamond \multimap \mathrm{Nat} \qquad\qquad \mathrm{S} : \diamond \multimap \mathrm{Nat} \multimap \mathrm{Nat}$$

- Iteration gives you them back:

$$\frac{\Gamma_1, d : \diamond \vdash M_z : A \qquad x : A, d : \diamond \vdash M_s : A \qquad \Gamma_2 \vdash N : \mathrm{Nat}}{\Gamma_1, \Gamma_2 \vdash \mathrm{iter}_A(d.M_z, x\ d.M_s, N) : A}$$

- Conservation of iterability

Iterating a function —

- Assume we have a function $f : A \multimap A$

  e.g., one step of a Turing machine

- Linear $\binom{n}{1}$ iterations:

  $$\mathrm{iter}(\cdots) : \mathrm{Nat} \otimes A \multimap \mathrm{Nat} \otimes A$$

  Reconstruct the Nat on the way through

- $\binom{n}{2}$ iterations:

  $$\mathrm{iter}(\cdots) : \mathrm{Nat} \otimes A \multimap \mathrm{Nat} \otimes A$$

  Reconstruct the Nat on the way through
  Do a nested linear iteration on the reconstructed number

- $\binom{n}{3}$ iterations: Iterate the above

Iterating a function —

- Obtain a $\binom{n}{k}$ iterator for any k
- And get the original number back!
- Chain them together to get any polynomial:

$$p(n) = \sum_{i=0}^{k} p_i \binom{n}{k}$$

- So we get polytime completeness

Amortised Resource Analysis — (Hofmann & Jost, POPL 2003)

- Reinterpret $\diamond$ as the cost of a step of iteration
- Inspired by Tarjan's *amortised complexity analyis*
  - storing potential inside data structures
- Building a Nat still requires $\diamond$s:

$$0 : \diamond \multimap \mathrm{Nat} \qquad\qquad \mathrm{S} : \diamond \multimap \mathrm{Nat} \multimap \mathrm{Nat}$$

- But iteration no longer gives you them back:

$$\frac{\Gamma_1 \vdash M_z : A \qquad x : A \vdash M_s : A \qquad \Gamma_2 \vdash N : \mathrm{Nat}}{\Gamma_1, \Gamma_2 \vdash \mathrm{iter}_A(M_z, x.M_s, N) : A}$$

- Back to linear time...

More flexibility

- Annotate data structures with number of $\diamond$s per constructor

$$\mathrm{Nat}^{p}$$

- Duplication:

$$\mathrm{Nat}^{p_1+p_2} \multimap \mathrm{Nat}^{p_1} \otimes \mathrm{Nat}^{p_2}$$

- Hofmann & Jost used linear programming to infer the ps

Regaining polynomial time — (Hoffmann & Hofmann, ESOP 2010)

- Annotate with sequences of naturals:

$$\mathrm{Nat}^{(p_1,\ldots,p_k)}$$

- Interpretation is that

$$\sum_{i=1}^{k} p_i \binom{n}{i}$$

number of $\diamond$s is attached to a natural n.

Regaining polynomial time — (Hoffmann & Hofmann, ESOP 2010)

- Annotate with sequences of naturals:

$$\mathrm{Nat}^{(p_1,\ldots,p_k)}$$

- Interpretation is that

$$\sum_{i=1}^{k} p_i \binom{n}{i}$$

number of $\Diamond$s is attached to a natural n.

- Iterator:

$$\frac{\Gamma_1 \vdash M_z : A \qquad n : \mathrm{Nat}^{(p_1+p_2, p_2+p_3, \ldots, p_k)}, d : \Diamond^{p_1}, x : A \vdash M_s : A \qquad \Gamma_2 \vdash N : \mathrm{Nat}^{(p_1+1,\ldots,p_k)}}{\Gamma_1, \Gamma_2 \vdash \mathrm{iter}(M_z, n\ d\ x.M_s, N) : A}$$

Adapting these systems to dependent types

Dependency and Accountancy

In Martin-Löf Type Theory

$$x_1 : S_1, \ldots, x_n : S_n \vdash M : T$$

In Martin-Löf Type Theory

$$x_1 : S_1, \ldots, x_n : S_n \vdash M : T$$

variables $x_1, \ldots, x_n$ are mixed usage

$$n : Nat, x : \mathsf{Fin}(n) \vdash x : \mathsf{Fin}(n)$$

$$n : Nat, x : \mathsf{Fin}(n) \vdash x : \mathsf{Fin}(n)$$

x is used *computationally*

$$n : Nat, x : \mathsf{Fin}(n) \vdash x : \mathsf{Fin}(n)$$

x is used *computationally*

n is used *logically*

In Linear Logic

$$x_1 : X_1, \ldots, x_n : X_n \vdash M : Y$$

In Linear Logic

$$x_1 : X_1, \ldots, x_n : X_n \vdash M : Y$$

the presence of a variable x records its usage
each x_i must be "used" by M exactly once

In Linear Logic

$$x_1 : X_1, \ldots, x_n : X_n \vdash M : Y$$

the presence of a variable x records its usage
each x_i must be "used" by M exactly once

Enables:

1. Insight into computational behaviour
2. e.g., time complexity

$$n : \mathsf{Nat}, x : \mathsf{Fin}(n) \vdash x : \mathsf{Fin}(n)$$

Can we read this judgement linearly?

$$n : \mathsf{Nat}, x : \mathsf{Fin}(n) \vdash x : \mathsf{Fin}(n)$$

Can we read this judgement linearly?

▷ n appears in the context, but is not used computationally

$$n : \mathsf{Nat}, x : \mathsf{Fin}(n) \vdash x : \mathsf{Fin}(n)$$

Can we read this judgement linearly?

- ▷ n appears in the context, but is not used computationally
- ▷ n appears *twice* in types

$$n : \mathsf{Nat}, x : \mathsf{Fin}(n) \vdash x : \mathsf{Fin}(n)$$

Can we read this judgement linearly?

- ▷ n appears in the context, but is not used computationally
- ▷ n appears *twice* in types

Is n even used at all?

$$n : \mathsf{Nat} \mid x : \mathsf{Fin}(n) \vdash x : \mathsf{Fin}(n)$$

$$n : \mathsf{Nat} \mid x : \mathsf{Fin}(n) \vdash x : \mathsf{Fin}(n)$$

- Separate *intuitionistic* / *unrestricted* uses and *linear* uses

$$n : \mathsf{Nat} \mid x : \mathsf{Fin}(n) \vdash x : \mathsf{Fin}(n)$$

- Separate *intuitionistic* / *unrestricted* uses and *linear* uses
- Types can depend on intuitionistic data, but not linear data

will come back to this...

$$n : \mathsf{Nat} \mid x : \mathsf{Fin}(n) \vdash x : \mathsf{Fin}(n)$$

- Separate *intuitionistic* / *unrestricted* uses and *linear* uses
- Types can depend on intuitionistic data, but not linear data

will come back to this...

(Barber, 1996)
(Cervesato and Pfenning, 2002)
(Krishnaswami, Pradic, and Benton, 2015)
(Vákár, 2015)

Separation interferes with dependency:

$$n : \mathsf{Nat} \mid x : \mathsf{Fin}(n) \vdash (x, \mathsf{refl}(x)) : (y : \mathsf{Fin}(n)) \times (x \equiv y)$$

Separation interferes with dependency:

$$n : \mathsf{Nat} \mid x : \mathsf{Fin}(n) \vdash (x, \mathsf{refl}(x)) : (y : \mathsf{Fin}(n)) \times (x \equiv y)$$

$$n : \mathsf{Nat}, x : \mathsf{Fin}(n) \mid \hat{x} : \hat{\mathsf{Fin}}(n, x) \vdash (x, \hat{x}, \mathsf{refl}(x)) : (y : \mathsf{Fin}(n)) \times \hat{\mathsf{Fin}}(n, y) \otimes (x \equiv y)$$

Quantitative Coeffect calculi:

$$x_1 \overset{\rho_1}{:} S_1, \ldots, x_n \overset{\rho_n}{:} S_n \vdash M : T$$

Quantitative Coeffect calculi:

$$x_1 \overset{\rho_1}{:} S_1, \ldots, x_n \overset{\rho_n}{:} S_n \vdash M : T$$

- The ρ_i record usage from some semiring R
  - $1 \in R$ — a use
  - $0 \in R$ — not used
  - $\rho_1 + \rho_2$ — adding up uses (e.g., in an application)
  - $\rho_1 \rho_2$ — nested uses

Quantitative Coeffect calculi:

$$x_1 \overset{\rho_1}{:} S_1, \ldots, x_n \overset{\rho_n}{:} S_n \vdash M : T$$

▷ The ρ_i record usage from some semiring R

- $1 \in R$ — a use
- $0 \in R$ — not used
- $\rho_1 + \rho_2$ — adding up uses (e.g., in an application)
- $\rho_1 \rho_2$ — nested uses

(Petricek, Orchard, and Mycroft, 2014)
(Brunel, Gaboardi, Mazza, and Zdancewic, 2014)
(Ghica and Smith, 2014)

Can we adapt this idea to dependent types?

Can we adapt this idea to dependent types?

McBride's idea:

- allow 0-usage data to appear in types.
  (McBride, 2016)

Can we adapt this idea to dependent types?

McBride's idea:

- allow 0-usage data to appear in types.

  (McBride, 2016)

$$x_1 \overset{\rho_1}{:} S_1, \ldots, x_n \overset{\rho_n}{:} S_n \vdash M \overset{\sigma}{:} T$$

where $\sigma \in \{0, 1\}$.

- $\sigma = 1$ — the "real" computational world
- $\sigma = 0$ — the types world

(allowing arbitrary ρ yields a system where substitution is inadmissible)

Can we adapt this idea to dependent types?

McBride's idea:

- allow 0-usage data to appear in types.
  (McBride, 2016)

$$x_1 \overset{\rho_1}{:} S_1, \ldots, x_n \overset{\rho_n}{:} S_n \vdash M \overset{\sigma}{:} T$$

where $\sigma \in \{0, 1\}$.

- $\sigma = 1$ — the "real" computational world
- $\sigma = 0$ — the types world

(allowing arbitrary ρ yields a system where substitution is inadmissible)

Zero-ing is an admissible rule: $\dfrac{\Gamma \vdash M \overset{1}{:} T}{0\Gamma \vdash M \overset{0}{:} T}$ allowing promotion to the type world.

Quantitative Type Theory

Contexts

$$\frac{}{\diamond \vdash}\textsc{Emp} \qquad \frac{\Gamma \vdash \qquad 0\Gamma \vdash S}{\Gamma, x \overset{\rho}{:} S \vdash}\textsc{Ext}$$

Contexts

$$\frac{}{\diamond \vdash}\ \textsc{Emp} \qquad \frac{\Gamma \vdash \qquad 0\Gamma \vdash S}{\Gamma, x \overset{\rho}{:} S \vdash}\ \textsc{Ext}$$

Types

$$0\Gamma \vdash S$$

Contexts

$$\frac{}{\diamond \vdash}\ \textsc{Emp} \qquad \frac{\Gamma \vdash \qquad 0\Gamma \vdash S}{\Gamma, x \overset{\rho}{:} S \vdash}\ \textsc{Ext}$$

Types

$$0\Gamma \vdash S$$

Terms

$$\frac{0\Gamma, x \overset{\sigma}{:} S, 0\Gamma' \vdash}{0\Gamma, x \overset{\sigma}{:} S, 0\Gamma' \vdash x \overset{\sigma}{:} S}\ \textsc{Var} \qquad \frac{\Gamma \vdash M \overset{\sigma}{:} S \qquad 0\Gamma \vdash S \equiv T}{\Gamma \vdash M \overset{\sigma}{:} T}\ \textsc{Conv}$$

Π-*type formation*

$$\frac{0\Gamma \vdash S \qquad 0\Gamma, x \overset{0}{:} S \vdash T}{0\Gamma \vdash (x \overset{\pi}{:} S) \longrightarrow T}$$

Π-type formation

$$\frac{0\Gamma \vdash S \qquad 0\Gamma, x \overset{0}{:} S \vdash T}{0\Gamma \vdash (x \overset{\pi}{:} S) \longrightarrow T}$$

Π-type introduction and elimination

$$\frac{\Gamma, x \overset{\sigma\pi}{:} S \vdash M \overset{\sigma}{:} T}{\Gamma \vdash \lambda x \overset{\pi}{:} S.M \overset{\sigma}{:} (x \overset{\pi}{:} S) \longrightarrow T}$$

$$\frac{\Gamma_1 \vdash M \overset{\sigma}{:} (x \overset{\pi}{:} S) \longrightarrow T \qquad \Gamma_2 \vdash N \overset{\sigma'}{:} S \qquad 0\Gamma_1 = 0\Gamma_2 \qquad \sigma' = 0 \Leftrightarrow (\pi = 0 \vee \sigma = 0)}{\Gamma_1 + \pi\Gamma_2 \vdash M\,N \overset{\sigma}{:} T[N/x]}$$

Integrating Resource Constraints into QTT

A suitable semiring for affine linearity?

- Carrier: $\{0, 1, \omega\}$
- Ordered: $\omega < 1 < 0$
- Operations:

$+$	0	1	ω
0	0	1	ω
1	1	ω	ω
ω	ω	ω	ω

$\cdot$	0	1	ω
0	0	0	0
1	0	1	ω
ω	0	ω	ω

- Would admit an unrestricted $!$ modality.

Strict resource counting

- Carrier: $\mathbb{N}$
- Ordered: $\cdots < 2 < 1 < 0$
- Operations: normal operations on $\mathbb{N}$

Diamonds —

$$\frac{\Gamma \vdash}{0\Gamma \vdash \diamond}\ \textsc{Ty-Dia} \qquad\qquad \frac{0\Gamma \vdash}{0\Gamma \vdash * \overset{0}{:} \diamond}\ \textsc{Tm-Dia}$$

— In the $\sigma = 0$ fragment, $\diamond$s are free.

LFPL —

- Natural number introduction

$$\frac{\Gamma \vdash d \overset{\sigma}{:} \diamond}{\Gamma \vdash \mathrm{zero}@d \overset{\sigma}{:} \mathrm{Nat}} \qquad \frac{\Gamma \vdash d \overset{\sigma}{:} \diamond \qquad \Gamma \vdash n \overset{\sigma}{:} \mathrm{Nat}}{\Gamma \vdash \mathrm{succ}(n)@d \overset{\sigma}{:} \mathrm{Nat}}$$

- Natural number elimination ($\sigma = 1$ case)

$$\frac{\begin{array}{c} 0\Gamma, x : \mathrm{Nat} \vdash A \qquad \Gamma_1, d \overset{1}{:} \diamond \vdash M_z \overset{1}{:} A\{\mathrm{zero}@* / x\} \\ d \overset{1}{:} \diamond, n \overset{0}{:} \mathrm{Nat}, r \overset{1}{:} A\{n/x\} \vdash M_s \overset{1}{:} A\{\mathrm{succ}(n)@* / x\} \qquad \Gamma_2 \vdash N \overset{1}{:} \mathrm{Nat} \qquad \Gamma_1 + \Gamma_2 = \Gamma \end{array}}{\Gamma \vdash \mathrm{iter}(x.A, d.M_z, d\ n\ r.M_s, N) : A\{N/x\}}$$

- Crucial: n is not available for compuational use in M_s.

Encoding lists

- Define (in $\sigma = 0$ fragment):

$$\mathrm{Vec}\ A : \mathrm{Nat} \longrightarrow \mathrm{Set}$$

  by iteration on the natural number.

- Lists:

$$\mathrm{List}\ A = (n \overset{1}{:} \mathrm{Nat}) \otimes \mathrm{Vec}\ A\ n$$

Amortised Analysis

- Unrestricted introduction rules for natural numbers:

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathrm{zero} \overset{\sigma}{:} \mathrm{Nat}} \qquad \frac{\Gamma \vdash N \overset{\sigma}{:} \mathrm{Nat}}{\Gamma \vdash \mathrm{succ}(N) \overset{\sigma}{:} \mathrm{Nat}}$$

- Postulate:

$$\Diamond^{(p_1,\ldots,p_k)} : \mathrm{Nat} \longrightarrow \mathrm{Set}$$

- with:

$$\mathrm{split} : (n \overset{0}{:} \mathrm{Nat}) \longrightarrow \Diamond^{(p_1+p'_1,\ldots,p_k+p'_k)}(n) \multimap \Diamond^{(p_1,\ldots,p_k)}(n) \otimes \Diamond^{(p'_1,\ldots,p'_k)}(n)$$

$$\mathrm{join} : (n \overset{0}{:} \mathrm{Nat}) \longrightarrow \Diamond^{(p_1,\ldots,p_k)}(n) \otimes \Diamond^{(p'_1,\ldots,p'_k)}(n) \multimap \Diamond^{(p_1+p'_1,\ldots,p_k+p'_k)}(n)$$

$$\mathrm{shift} : (n \overset{0}{:} \mathrm{Nat}) \longrightarrow \Diamond^{(p_1,\ldots,p_k)}(\mathrm{succ}(n)) \multimap \Diamond^{(p_1+p_2,\ldots,p_k)}(n)$$

Amortised Analysis

- Natural number elimination ($\sigma = 1$ case)

$$\frac{0\Gamma, x : \mathrm{Nat} \vdash A \qquad \Gamma_1 \vdash M_z \overset{1}{:} A\{\mathrm{zero}/x\} \qquad n \overset{1}{:} \mathrm{Nat}, r \overset{1}{:} A\{n/x\} \vdash M_s \overset{1}{:} A\{\mathrm{succ}(n)/x\} \qquad \Gamma_2 \vdash N \overset{1}{:} \mathrm{Nat} \qquad \Gamma_3 \vdash D \overset{1}{:} \Diamond^{(1)}(N) \qquad \Gamma_1 + \Gamma_2 + \Gamma_3 = \Gamma}{\Gamma \vdash \mathrm{iter}(x.A, M_z, n\ r.M_s, N) : A\{N/x\}}$$

- n is available for use in M_s
- Pay up front for the iteration with D
- Get nested iteration by passing in enough $\Diamond$s to pay for it

$$A[n] = \Diamond^{(p_1, \ldots, p_k)}(n) \multimap B[n]$$

Semantic Interpretation : Soundness

1. Category $\mathcal{L}$ for interpreting contexts and simultaneous substitutions
   Computational and extensional content
2. Category $\mathcal{C}$ for intepretation contexts and simultaneous substitutions
   Only extensional content

1. Category $\mathcal{L}$ for interpreting contexts and simultaneous substitutions
   Computational and extensional content
2. Category C for intepretation contexts and simultaneous substitutions
   Only extensional content
3. $U : \mathcal{L} \to C$ forgetting the computational content;

Quantitative Category with Families

1. Category $\mathcal{L}$ for interpreting contexts and simultaneous substitutions
   Computational and extensional content
2. Category C for intepretation contexts and simultaneous substitutions
   Only extensional content
3. $U : \mathcal{L} \to C$ forgetting the computational content;
4. Addition and scaling structure on $\mathcal{L}$, fibred over U;
   Can add Γ_1 and Γ_2 if $U\Gamma_1 = U\Gamma_2$

Quantitative Category with Families

1. Category $\mathcal{L}$ for interpreting contexts and simultaneous substitutions
   Computational and extensional content
2. Category $\mathcal{C}$ for intepretation contexts and simultaneous substitutions
   Only extensional content
3. $U : \mathcal{L} \to \mathcal{C}$ forgetting the computational content;
4. Addition and scaling structure on $\mathcal{L}$, fibred over U;
   Can add Γ_1 and Γ_2 if $U\Gamma_1 = U\Gamma_2$
5. Semantic types formed with respect to $\mathcal{C}$:

$$S \in \mathrm{Ty}(\Delta), \qquad \Delta \in \mathrm{Ob}\mathcal{C};$$

Quantitative Category with Families

1. Category $\mathcal{L}$ for interpreting contexts and simultaneous substitutions

   Computational and extensional content

2. Category $\mathcal{C}$ for intepretation contexts and simultaneous substitutions

   Only extensional content

3. $U : \mathcal{L} \to \mathcal{C}$ forgetting the computational content;
4. Addition and scaling structure on $\mathcal{L}$, fibred over U;

   Can add Γ_1 and Γ_2 if $U\Gamma_1 = U\Gamma_2$

5. Semantic types formed with respect to $\mathcal{C}$:

$$S \in \mathrm{Ty}(\Delta), \qquad \Delta \in \mathrm{Ob}\mathcal{C};$$

6. Semantic terms, resourced and unresourced:

$$\begin{aligned} M &\in \mathrm{Tm}(\Delta, S), & \Delta &\in \mathrm{Ob}\mathcal{C}, S \in \mathrm{Ty}(\Delta) \\ M &\in \mathrm{RTm}(\Gamma, S), & \Gamma &\in \mathrm{Ob}\mathcal{L}, S \in \mathrm{Ty}(U\Gamma); \end{aligned}$$

Quantitative Category with Families

1. Category $\mathcal{L}$ for interpreting contexts and simultaneous substitutions
   Computational and extensional content
2. Category $\mathcal{C}$ for intepretation contexts and simultaneous substitutions
   Only extensional content
3. $U : \mathcal{L} \to \mathcal{C}$ forgetting the computational content;
4. Addition and scaling structure on $\mathcal{L}$, fibred over U;
   Can add Γ_1 and Γ_2 if $U\Gamma_1 = U\Gamma_2$
5. Semantic types formed with respect to $\mathcal{C}$:

$$S \in \mathrm{Ty}(\Delta), \qquad \Delta \in \mathrm{Ob}\mathcal{C};$$

6. Semantic terms, resourced and unresourced:

$$\begin{aligned} M &\in \mathrm{Tm}(\Delta, S), & \Delta &\in \mathrm{Ob}\mathcal{C}, S \in \mathrm{Ty}(\Delta) \\ M &\in \mathrm{RTm}(\Gamma, S), & \Gamma &\in \mathrm{Ob}\mathcal{L}, S \in \mathrm{Ty}(U\Gamma); \end{aligned}$$

7. Semantic zero-ing: $U : \mathrm{RTm}(\Gamma, S) \to \mathrm{Tm}(U\Gamma, S)$;

Quantitative Category with Families

1. Category $\mathcal{L}$ for interpreting contexts and simultaneous substitutions

   Computational and extensional content

2. Category C for intepretation contexts and simultaneous substitutions

   Only extensional content

3. $U : \mathcal{L} \to C$ forgetting the computational content;

4. Addition and scaling structure on $\mathcal{L}$, fibred over U;

   Can add Γ_1 and Γ_2 if $U\Gamma_1 = U\Gamma_2$

5. Semantic types formed with respect to C:

$$S \in \mathrm{Ty}(\Delta), \qquad \Delta \in \mathrm{Ob}C;$$

6. Semantic terms, resourced and unresourced:

$$\begin{aligned} M &\in \mathrm{Tm}(\Delta, S), & \Delta &\in \mathrm{Ob}C, S \in \mathrm{Ty}(\Delta) \\ M &\in \mathrm{RTm}(\Gamma, S), & \Gamma &\in \mathrm{Ob}\mathcal{L}, S \in \mathrm{Ty}(U\Gamma); \end{aligned}$$

7. Semantic zero-ing: $U : \mathrm{RTm}(\Gamma, S) \to \mathrm{Tm}(U\Gamma, S)$;

8. Resourced counterparts of substitution and comprehension, preserved by U.

1. Let $\mathcal{A}$ be an $\mathcal{R}$-LCA
2. Let $\mathcal{C}$ = Set, category of sets and functions
3. Let $\mathcal{L}$ be:
   - Objects: $(|\Gamma|, \models_\Gamma)$, where $|\Gamma|$ is a set, and $\models_\Gamma \subseteq \mathcal{A} \times |\Gamma|$;
   - Morphisms: $f : |\Gamma_1| \to |\Gamma_2|$, for which
     there exists an $a_f \in \mathcal{A}$ such that for all x, a_x, $a_x \models_{\Gamma_1} x$ implies $a_f \cdot a_x \models f(x)$.

   sets with computational information
4. $U : \mathcal{L} \to \mathcal{C}$ forgets the computational information
5. Types $S \in \mathrm{Ty}(\Delta)$ include computational information, but:
   only depend on non-computational part
6. Terms $M \in \mathrm{RTm}(\Gamma, S)$ are tracked by realisers from $\mathcal{A}$
7. Terms $M \in \mathrm{Tm}(\Delta, S)$ are set theoretic functions

Read constructively, yields an "efficient" compilation method for QTT, which respects and uses the usage information.

"Length" models ~(Hofmann, 2003)

- $\mathcal{A} = \mathbb{N}$
- Application:

$$a \cdot b = a + b$$

- Combinators:

$$\begin{aligned} B &= 0 \\ C &= 0 \\ I &= 0 \end{aligned}$$

- "Precious" booleans:

$$\begin{aligned} \diamond &= 1 \\ T &= 1 \\ F &= 1 \end{aligned}$$

- Have to restrict interpretation of terms to always be 0

Realisability for ICC (Dal Lago & Hofmann, 2011)

- $\mathbb{N}_{-\infty}$ a category with objects $\mathbb{N} \cup \{-\infty\}$ and $m \to n$ if $m \leq n$, with $-\infty \leq n$
  - Strict symmetric monoidal category with $(+, 0)$
- A resource monoid M is a $\mathbb{N}_{-\infty}$ enriched strict symmetric monoidal category.
- Assume a model of computation with a cost model:

$$e, \eta \Downarrow_k v$$

  Expressions $\mathcal{E}$ and values $\mathcal{V}$.
- Realisers are pairs $\alpha, v \in M \times \mathcal{V}$
- Functions $f : \Gamma_1 \to \Gamma_2$:
  - $f : |\Gamma_1| \to |\Gamma_2|$
  - $e \in \mathcal{E}$, $\gamma \in M$
  - for all α, v, a. $\alpha, v \models_{\Gamma_1} a$ implies
    exists β, k, v' s.t. $e, [v] \Downarrow_k v'$ and $\beta, v' \models_{\Gamma_2} f(a)$ and $k \leq M(\alpha + \gamma, \beta)$

Resource Monoids

- For LFPL: $M \ni (n, f)$, where
  - $n \in \mathbb{N}$ is the amount of iterability (number of $\diamond$s)
  - f is a polynomial with $\mathbb{N}$ coefficients
  - Cost differencing:

$$M((n, f), (m, g)) = \begin{cases} g(m) - f(m) & n \leq m \text{ and } (g - f) \text{ is non-negative and non-decreasing } \geq m \\ -\infty & \text{otherwise} \end{cases}$$

- $(n, 0), * \models_{\diamond} * \text{ if } n \geq 1.$

Resource Monoids

- For LFPL: $M \ni (n, f)$, where
  - $n \in \mathbb{N}$ is the amount of iterability (number of $\Diamond$s)
  - f is a polynomial with $\mathbb{N}$ coefficients
  - Cost differencing:

$$M((n,f),(m,g)) = \begin{cases} g(m) - f(m) & n \leq m \text{ and } (g - f) \text{ is non-negative and non-decreasing } \geq m \\ -\infty & \text{otherwise} \end{cases}$$

- $(n, 0), {}^{*} \models_{\Diamond} * \text{ if } n \geq 1.$
- For Amortised Analysis: restrict f to be degree ≤ 1
- Cost is accounted for directly by the $\Diamond$s

Internalising Resource Reasoning?

Intuition:

$$\frac{0\Gamma \vdash A}{0\Gamma \vdash R(A)}$$

is the type of "efficiently" realisable programs of type A.

For LFPL: $R(\mathrm{Nat} \multimap \mathrm{Nat}^{\circ}) = $ Poly time functions

Intuition:

$$\frac{0\Gamma \vdash A}{0\Gamma \vdash R(A)}$$

is the type of "efficiently" realisable programs of type A.

For LFPL: $R(\mathrm{Nat} \multimap \mathrm{Nat}^{\circ}) = $ Poly time functions

Introduction and Elimination

$$\frac{0\Gamma \vdash a \overset{1}{:} A}{0\Gamma \vdash r(a) \overset{\sigma}{:} R(A)} \qquad \frac{\Gamma \vdash a \overset{\sigma}{:} R(A)}{\Gamma \vdash r^{-1}(a) \overset{\sigma}{:} A}$$

Intuition:

$$\frac{0\Gamma \vdash A}{0\Gamma \vdash R(A)}$$

is the type of "efficiently" realisable programs of type A.

For LFPL: $R(\mathrm{Nat} \multimap \mathrm{Nat}^{\circ}) = $ Poly time functions

Introduction and Elimination

$$\frac{0\Gamma \vdash a \overset{1}{:} A}{0\Gamma \vdash r(a) \overset{\sigma}{:} R(A)} \qquad \frac{\Gamma \vdash a \overset{\sigma}{:} R(A)}{\Gamma \vdash r^{-1}(a) \overset{\sigma}{:} A}$$

E.g.

$$\frac{0\Gamma \vdash M \overset{1}{:} (x \overset{1}{:} A) \to B}{0\Gamma \vdash r(M) \overset{0}{:} R((x \overset{1}{:} A) \to B)}$$

means "the function M is realisable in the underlying computational model".

Interpretation:

$$|[\![R(A)]\!]\gamma| = \{a \in |[\![A]\!]\gamma| \mid \exists x \in \mathcal{A}.\ x \models_{A(\gamma)} a\}$$

Interpretation:

$$|[\![R(A)]\!]\gamma| = \{a \in |[\![A]\!]\gamma| \mid \exists x \in \mathcal{A}.\ x \models_{A(\gamma)} a\}$$

Internalise the realisability:

$$R(A) = \exists a \overset{0}{:} A.\exists p \overset{0}{:} \mathsf{Prog}.p \models_A a$$

with definition of $\models_A$ above, capture polytime functions as a type.

Interpretation:

$$|[\![R(A)]\!]\gamma| = \{a \in |[\![A]\!]\gamma| \mid \exists x \in \mathcal{A}.\ x \models_{A(\gamma)} a\}$$

Internalise the realisability:

$$R(A) = \exists a \overset{0}{:} A. \exists p \overset{0}{:} \mathsf{Prog}. p \models_A a$$

with definition of $\models_A$ above, capture polytime functions as a type.

Note: $R(A)$ must be a proposition – cannot allow intensional information to be used to make decisions in the extensional world!

Summary

- Quantitative Type Theory:
  Fixed and extended formulation of McBride's "Plenty o' Nuttin' " system
- Quantitative Type Theory for ICC
  Dependently typed version of LFPL and Amortised Analysis
- Categorical and Realisability models
- Goal: to combine resource-ful and resource-less programming and reasoning.

- Quantitative Type Theory:
  Fixed and extended formulation of McBride's "Plenty o' Nuttin' " system

- Quantitative Type Theory for ICC
  Dependently typed version of LFPL and Amortised Analysis

- Categorical and Realisability models

- Goal: to combine resource-ful and resource-less programming and reasoning.

Related Work

- Sized types
  Used for controlling well foundedness
  For complexity analysis require "tick" monads
- Gaboardi and Dal Algo: Linear Dependent Types for ICC
  Dependent Types only for counting time
- Future:
  - Soft linear logic, LAL, EAL
  - Polytime mathematics?