Resourceful Dependent Types

Andreas Abel[1]

[1]Department of Computer Science and Engineering
Chalmers and Gothenburg University, Sweden

Types for Proofs and Programs
TYPES 2018
University Do Minho, Braga, Portugal
20 June 2018

Martin Hofmann's Resourceful Types

- CSL 1997

  A mixed modal/linear lambda calculus with applications to Bellantoni-Cook safe recursion

- ESOP 2000

  A type system for bounded space and functional in-place update

- POPL 2003, with S. Jost

  Static prediction of heap space usage for first-order functional programs

- Projects: MRG, Embounded, ...

Martin Hofmann's Breakthroughs on Dependent Types

- LiCS 1994, with T. Streicher

  The Groupoid Model Refutes Uniqueness of Identity Proofs

- TYPES 1995

  Conservativity of Equality Reflection over Intensional Type Theory

- Distinguished dissertation 1997

  Extensional constructs in intensional type theory
  Syntax and semantics of dependent types

What is a linear function?

- Which functions should be considered *linear*?

$$\begin{array}{lcl} \mathsf{dup} & : & \mathbb{N} \to \mathbb{N} \times \mathbb{N} \\ \mathsf{dup}\ n & = & (n, n) \end{array}$$

- Is dup linear?

Linear λ-definability

- Consider a universe of types Ty with $(\!|\_|\!) : \mathsf{Ty} \to \mathsf{Set}$.
- A function $f : (\!|A|\!)$ is *$\mathcal{X}$-definable* if there exists a closed term $\vdash t : A$ in calculus $\mathcal{X}$ such that $(\!|t|\!) = f$.
- "dup linear" depends on $\mathcal{X}$:
  - dup not definable in linear STLC.

$$\begin{array}{lcl} \mathsf{dup} & : & \mathbb{N} \multimap \mathbb{N} \otimes \mathbb{N} \\ \mathsf{dup}\ n & = & (n, ?) \end{array}$$

  - dup definable in linear Gödel's T.

$$\begin{array}{lcl} \mathsf{dup} & : & \mathbb{N} \multimap \mathbb{N} \otimes \mathbb{N} \\ \mathsf{dup}\ \mathsf{zero} & = & (\mathsf{zero}, \mathsf{zero}) \\ \mathsf{dup}\ (\mathsf{suc}\ n) & = & \mathsf{suc}_2\ (\mathsf{dup}\ n) \end{array}$$

$$\begin{array}{lcl} \mathsf{suc}_2 & : & \mathbb{N} \otimes \mathbb{N} \multimap \mathbb{N} \otimes \mathbb{N} \\ \mathsf{suc}_2\ (n, m) & = & (\mathsf{suc}\ n,\ \mathsf{suc}\ m) \end{array}$$

A Free Theorem from linear typing

Theorem (Bob Atkey)

Given an abstract type K of "keys" with operation

$$\mathsf{compare} : (K \otimes K) \multimap (\mathsf{Bool} \otimes K \otimes K)$$

and a program (i.e., closed term)

$$f : \mathsf{List}\, K \multimap \mathsf{List}\, K$$

then f is a list permutation.

Proof formalized in Agda.

`https://github.com/bobatkey/sorting-types`. □

Proof of the free theorem

- Category $\mathbb{W}$ of lists over K and permutations $w \hookrightarrow w'$.
- $\mathbb{W}$ symmetric monoidal: empty list $\mathbb{1}$, concatenation $\otimes$.
- Logical relation $\models_A \subseteq \mathbb{W} \times A$ natural in $\mathbb{W}$ (i.e., closed under $\hookrightarrow$).
- $w \models_A a$: *value a can be constructed exactly from the resources w.*

$$
\begin{array}{lll}
w \models_1 () & \text{iff } w = \mathbb{1} \\
w \models_{A_1 \oplus A_2} \mathsf{in}_i(a) & \text{iff } w \models_{A_i} a \\
w \models_{A \otimes B} (a, b) & \text{iff } w \hookrightarrow w_1 \otimes w_2 \text{ and } w_1 \models_A a \text{ and } w_2 \models_B b \\
 & \qquad \text{for some } w_1, w_2 \\
w \models_{A \multimap B} f & \text{iff } w' \models_A a \text{ implies } w \otimes w' \models_B f(a) \text{ for all } w'
\end{array}
$$

- Setting: $w \models_K k$ iff w is singleton k.
- Remember: $\mathsf{List}\, K = 1 \oplus (K \otimes \mathsf{List}\, K)$.
- Consequence: $w \models_{\mathsf{List}\, K} ks$ iff w is a permutation of ks.

Proof of the free theorem (ctd.)

- Fundamental theorem: If $\Gamma \vdash t : A$ and $w \models_\Gamma \sigma$ then $w \models_A t\sigma$.
- $\vdash f : \mathsf{List}\, K \multimap \mathsf{List}\, K$ implies $\mathbb{1} \models_{\mathsf{List}\, K \multimap \mathsf{List}\, K} f$
- With $ks \models_{\mathsf{List}\, K} ks$ have $\mathbb{1} \otimes ks \models f(ks)$, thus $ks \hookrightarrow f(ks)$.

Remarks:

- We call the world w of (mandatorily) consumable resources *support*.
- Elements of *closed* types (not mentioning K) have *empty* support.
- Eliminators like $\mathsf{if} : \mathsf{Bool} \multimap (A \& A) \multimap A$ use additive conjunction $\&$.

$$w \models_{A\&B} (a, b) \quad \text{iff} \quad w \models_A a \text{ and } w \models_B b$$

- Subexponentials for $n \in \mathbb{N}$ where $w^n = w \otimes \ldots \otimes w$ (n times):

$$\begin{array}{lll} w \models_{!^n A} a & \text{iff} & w \hookrightarrow {w_0}^n \text{ and } w_0 \models_A a \text{ for some } w_0 \\ w \models_{?^n A} a & \text{iff} & w^n \models_A a \end{array}$$

- Gives quadratic functions like $\lambda^2 x.\, (x, x) : !^2 A \multimap A \times A$. But affine?

Choice of resources

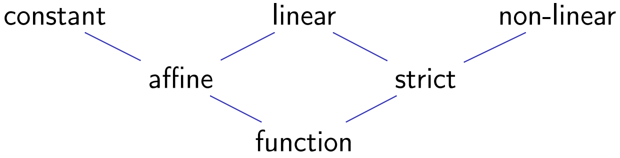
- Abstract K with $e : K$ and $\_ \cdot \_ : K \multimap K \multimap K$ and boolean $b : B$:

$$\begin{array}{lcl} \lambda^{\{0,1\}}x.\ \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ e & : & !^{\{0,1\}}K \multimap K \\ \lambda^{\{1,2\}}x.\ \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ x \cdot x & : & !^{\{1,2\}}K \multimap K \end{array}$$

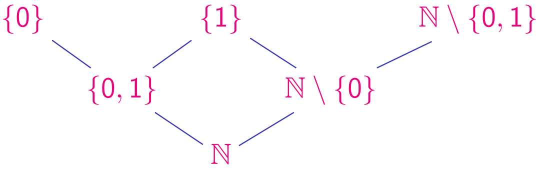
  Imprecision in usage quantity of x.
- Want $!^q A \multimap B$ for $q \subseteq \mathbb{N}$.
- Extend $\mathbb{W}$ by non-empty additive products $\&_{i \in q} A_i$ (infima).
- Morphisms $w \hookrightarrow w'$ now include dropping of alternatives $A \,\&\, B \hookrightarrow A$. In general, $\&_{i \in q} A_i \hookrightarrow \&_{j \in q'} A_j$ for $q' \subseteq q$.
- Exponent: $w^q = \&_{n \in q} w^n$.
- $w_1 \models_{!^q A} a$ iff $w_2 \models_A a$ for some w_2 with $w_1 \hookrightarrow {w_2}^q$.
- Ordinary $A \to B$ is $!^{\mathbb{N}} A \multimap B$.

Quantity lattice

- Function classification:



- Expressed as quantitative information $q \subseteq \mathbb{N}$ in $(!^q A) \multimap B$:



- Call this lattice $\mathbb{Q}$.

Quantity semiring

- Composition:

$$f : !^q B \multimap C \quad \text{and} \quad g : !^r A \multimap B \quad \text{implies} \quad f \circ g : !^{q \cdot r} A \multimap C$$

- Multiplication $q \cdot r = \{m \cdot n \mid m \in q, n \in r\}$ $\boxed{\text{rounded up}}$ to be in $\mathbb{Q}$.
- Choice:

$$u : !^q A \quad \text{and} \quad v : !^r A \quad \text{implies} \quad \text{if } x \text{ then } u \text{ else } v : !^{q+r} A$$

- Addition $q + r = \{m + n \mid m \in q, n \in r\}$ rounded up to be in $\mathbb{Q}$.

Dependent linear types

- Multiplicative linear dependent function and pair types.

$$w \models_{\Pi\, A\, F} f \quad \text{iff} \quad w' \models_A a \text{ implies } w \otimes w' \models_{F(a)} f(a) \text{ for all } w'$$

$$w \models_{\Sigma\, A\, F} (a, b) \quad \text{iff} \quad w_1 \models_A a \text{ and } w_2 \models_{F(a)} b \text{ for some } w_1, w_2 \text{ with } w \hookrightarrow w_1 \otimes w_2$$

- Obvious, no?

Dependent linear types, what took you so long?

- 1972: Martin-Löf: (Dependent) Type Theory
- 1987: Girard: Linear logic
- *(3 decades later)*
- 2016: McBride: I got plenty of nuttin'
- 2018: Atkey: Syntax and Semantics of Quantitative Type Theory
- What took us so long?
- (Wrong) paradigms!?
  - Focus on structural rules (weakening, contraction)!?
  - Separate contexts for linear and intuitionistic assumptions!?
  - Same quantity context for term and types!?

    $\Gamma \vdash t : A$ implies $\Gamma \vdash A : \mathsf{Type}$

  - Specific models of linearity!?
  - Missing generalization to quantitative typing!?

Quantitative type theory

- Syntax ($q, r \in \mathsf{Q}$):

$$
\begin{array}{rcll}
t, u, A, F & ::= & x & \text{name (free variable)} \\
& | & \lambda^q x.\, t & \lambda\text{-abstraction (binder) with quantity} \\
& | & t \cdot^q u & \text{application with quantity} \\
& | & \Pi^{q,r} A\, F & \text{dependent function type (no binder)} \\
& | & \mathsf{U}_\ell & \text{sort}
\end{array}
$$

- Usage calculation $\boxed{|t| : \mathsf{Var} \to \mathsf{Q}}$.

$$
\begin{array}{rcl}
|x| & = & \{x \mapsto 1\} \\
|t \cdot^q u| & = & |t| + q|u| \\
|\lambda^q x.\, t| & = & |t| \setminus x \\
|\mathsf{U}_\ell| & = & \emptyset \\
|\Pi^{q,r} A\, F| & = & |A| + |F|
\end{array}
$$

Quantitative typing

$$\frac{\vdash \Gamma}{\Gamma \vdash x : \Gamma(x)} \qquad \frac{\Gamma \vdash t : \Pi^{q,r} A\, F \qquad \Gamma \vdash u : A}{\Gamma \vdash t \cdot^{q} u : F \cdot^{r} u}$$

$$\frac{\Gamma, x{:}A \vdash t : F \cdot^{r} x}{\Gamma \vdash \lambda^{q} x.\, t : \Pi^{q,r} A\, F}\ q \supseteq |t|^{x}$$

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathsf{U}_{\ell} : \mathsf{U}_{\ell'}}\ \ell{<}\ell' \qquad \frac{\Gamma \vdash A : \mathsf{U}_{\ell} \qquad \Gamma \vdash F : A \xrightarrow{r} \mathsf{U}_{\ell}}{\Gamma \vdash \Pi^{q,r} A\, F : \mathsf{U}_{\ell}}$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash A \leq B}{\Gamma \vdash t : B}$$

Conclusions

- Quantitative typing generalizes linear typing.
- Practical uses:
  - Cardinality analysis in compilers: strictness, dead code.
  - Differential privacy (Reed Peirce ICFP 2010)
  - Erasure in type theory (EPTS).
  - Security typing!
- Thesis:

  The generalization of linear typing to quantitative typing allows a smooth integration with dependent typing.

Related Work

- Simple types: abundance of quantitative type systems (TYPES 2015).
- McBride 2016: $\mathsf{Q} = \{\{0\}, \{1\}, \mathbb{N}\}$. Usage in types does not count!
- Atkey 2018, QTT: Q semiring.
- Brady: implementing McBride/Atkey system in Idris 2.

Future work

- CwF-like model for my variant of QTT.
- Internalize free theorems from linearity?!
- Relate to other modal type theories.
- Add to Agda.

Subtyping

$$\frac{\Gamma \vdash A = A' : \mathsf{U}_\ell}{\Gamma \vdash A \leq A'}$$

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathsf{U}_\ell \leq \mathsf{U}_{\ell'}}\ \ell \leq \ell'$$

$$\frac{\Gamma \vdash A' \leq A \qquad \Gamma, x{:}A' \vdash F \cdot^{r} x \leq F' \cdot^{r} x}{\Gamma \vdash \Pi^{q,r} A\, F \leq \Pi^{q,r} A'\, F'}$$