Resourceful Dependent Types

Andreas Abel

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*?

  
  \[
  \text{dup} : \mathbb{N} \rightarrow \mathbb{N} \times \mathbb{N} \\
  \text{dup } n \quad = \quad (n, n)
  \]

- Is \text{dup} linear?
Linear $\lambda$-definability

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

\[
\text{dup} : \mathbb{N} \to \mathbb{N} \otimes \mathbb{N} \\
\text{dup } n = (n, ?)
\]

- $\text{dup}$ definable in linear Gödel's T.

\[
\begin{align*}
\text{dup} & : \mathbb{N} \to \mathbb{N} \otimes \mathbb{N} \\
\text{dup zero} & = (\text{zero}, \text{zero}) \\
\text{dup (suc } n) & = \text{suc}_2 (\text{dup } n)
\end{align*}
\]

\[
\begin{align*}
\text{suc}_2 & : \mathbb{N} \otimes \mathbb{N} \to \mathbb{N} \otimes \mathbb{N} \\
\text{suc}_2 (n, m) & = (\text{suc } n, \text{suc } m)
\end{align*}
\]
A Free Theorem from linear typing

Theorem (Bob Atkey)

Given an abstract type $K$ of “keys” with operation

$$\text{compare} : (K \otimes K) \to (\text{Bool} \otimes K \otimes K)$$

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

$$f : \text{List } K \to \text{List } K$$

then $f$ is a list permutation.

Proof formalized in Agda.

Proof of the free theorem

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

\[
\begin{align*}
w \models_1 () & \quad \text{iff } w = \mathbf{1} \\
w \models_{A_1 \oplus A_2} \text{in}_i(a) & \quad \text{iff } w \models_{A_i} a \\
w \models_{A \otimes B} (a, b) & \quad \text{iff } w \leftrightarrow w_1 \otimes w_2 \text{ and } w_1 \models_A a \text{ and } w_2 \models_B b \\
& \quad \text{for some } w_1, w_2 \\
w \models_{A \rightarrow B} f & \quad \text{iff } w' \models_A a \text{ implies } w \otimes w' \models_B f(a) \text{ for all } w'
\end{align*}
\]

- Setting: $w \models_K k$ iff $w$ is singleton $k$.
- Remember: $\text{List } K = 1 \oplus (K \otimes \text{List } K)$.
- Consequence: $w \models_{\text{List } K} ks$ 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 $w \models_{A} t\sigma$.
- $\vdash f : \text{List } K \to \text{List } K$ implies $1 \models_{\text{List } K \to \text{List } K} f$
- With $ks \models_{\text{List } K} ks$ have $1 \otimes ks \models f(ks)$, 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 $\text{if} : \text{Bool} \to (A \& A) \to A$ 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):

\[ w \models !_n A a \quad \text{iff} \quad w \hookrightarrow w_0^n \text{ and } w_0 \models_{A} a \text{ for some } w_0 \]
\[ w \models !_n A a \quad \text{iff} \quad w^n \models_{A} a \]

- Gives quadratic functions like $\lambda^2 x. (x, x) : !_2 A \to A \times A$. But affine?
Choice of resources

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

$$
\lambda^{\{0,1\}} x. \text{if } b \text{ then } x \text{ else } e : \!\{0,1\} K \rightarrow K
$$
$$
\lambda^{\{1,2\}} x. \text{if } b \text{ then } x \text{ else } x \cdot x : \!\{1,2\} K \rightarrow K
$$

Imprecision in usage quantity of $x$.

- Want $!^q A \rightarrow B$ for $q \subseteq \mathbb{N}$.
- Extend $W$ by non-empty additive products $\&_{i \in q} A_i$ (infima).
- Morphisms $w \leftrightarrow w'$ now include dropping of alternatives $A \& B \leftrightarrow A$. In general, $\&_{i \in q} A_i \leftrightarrow \&_{j \in q'} A_j$ 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 $w_1 \leftrightarrow w_2^q$
- Ordinary $A \rightarrow B$ is $!^N A \rightarrow B$. 
Quantity lattice

- Function classification:
  - constant
  - linear
  - non-linear
  - affine
  - strict

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

- Call this lattice $Q$. 
Quantity semiring

- Composition:
  \[ f : !^q B \rightarrow C \quad \text{and} \quad g : !^r A \rightarrow B \quad \text{implies} \quad f \circ g : !^{q \cdot r} A \rightarrow C \]

- Multiplication \( q \cdot r = \{ m \cdot n \mid m \in q, n \in r \} \) rounded up to be in \( Q \).

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

- Addition \( q + r = \{ m + n \mid m \in q, n \in r \} \) rounded up to be in \( Q \).
Dependent linear types

- Multiplicative linear dependent function and pair types.

\[ w \vdash_{\Pi} A F \ f \quad \text{iff} \quad w' \vdash_{A} a \ \text{implies} \ w \otimes w' \vdash_{F(a)} f(a) \ \text{for all} \ w' \]

\[ w \vdash_{\Sigma} A F (a, b) \quad \text{iff} \quad w_1 \vdash_{A} a \ \text{and} \ w_2 \vdash_{F(a)} b \ \text{for some} \ w_1, w_2 \]
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!?

\[ \Gamma \vdash t : A \text{ implies } \Gamma \vdash A : \text{Type} \]

Specific models of linearity!? 
Missing generalization to quantitative typing!?
Quantitative type theory

- **Syntax** \((q, r \in \mathbb{Q})\):

  \[
  t, u, A, F \quad ::= \quad x \quad \text{name (free variable)} \\
  | \quad \lambda^q x. \ t \quad \text{\(\lambda\)-abstraction (binder) with quantity} \\
  | \quad t \cdot^q u \quad \text{application with quantity} \\
  | \quad \Pi^{q,r} A F \quad \text{dependent function type (no binder)} \\
  | \quad U_\ell \quad \text{sort}
  \]

- **Usage calculation** \(|t| : \text{Var} \to \mathbb{Q}|.

  \[
  \begin{align*}
  |x| &= \{x \mapsto 1\} \\
  |t \cdot^q u| &= |t| + |q| u| \\
  |\lambda^q x. t| &= |t| \setminus x \\
  |U_\ell| &= \emptyset \\
  |\Pi^{q,r} A F| &= |A| + |F|
  \end{align*}
  \]
Quantitative typing

\[ \Gamma \vdash x : \Gamma(x) \]

\[ \Gamma \vdash t : \Pi_{q,r} A F \quad \Gamma \vdash u : A \]

\[ \Gamma \vdash t \cdot q u : F \cdot r u \]

\[ \Gamma, x:A \vdash t : F \cdot r x \]

\[ \Gamma \vdash \lambda^q x. t : \Pi_{q,r} A F \]

\[ q \trianglerighteq |t|^x \]

\[ \Gamma \vdash \ell < \ell' \]

\[ \Gamma \vdash U_{\ell} : U_{\ell'} \]

\[ \Gamma \vdash A : U_{\ell} \quad \Gamma \vdash F : A \rightarrow U_{\ell} \]

\[ \Gamma \vdash \Pi_{q,r} A F : U_{\ell} \]

\[ \Gamma \vdash t : A \quad \Gamma \vdash A \leq B \]

\[ \Gamma \vdash t : B \]
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: $Q = \{\{0\}, \{1\}, \mathbb{N}\}$. 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

\[
\Gamma \vdash A = A' : U_\ell \\
\therefore \quad \Gamma \vdash A \leq A'
\]

\[
\Gamma \vdash \ell \leq \ell'
\]

\[
\Gamma \vdash U_\ell \leq U_{\ell'}
\]

\[
\Gamma \vdash A' \leq A \\
\Gamma, \ x: A' \vdash F \cdot {}^r x \leq F' \cdot {}^r x
\]

\[
\Gamma \vdash \Pi^{q,r} A F \leq \Pi^{q,r} A' F'
\]