Adventures in Two-Dimensional Type Theory

Robert Harper
with Daniel R. Licata
(thanks to Steve Awodey and Peter Lumsdaine)

March, 2011

Background

Computation with syntax, including binding and scope.

- Integrate derivability with admissibility in a logical framework.
- Integrate higher-order abstract syntax with computation on syntax.
- cf Beluga, Delphin, and their derivatives.

Earlier work focused on polarity:

- Positive function space: derivability/hoas/templates.
- Negative function space: admissibility/computation.
- cf Andreoli, Girard, Zeilberger.

Background

Derivability: $J_1, \ldots, J_n \vdash J$.

- Is J derivable, given $J_1, \ldots, J_n$ as axioms?
- Not implication! (Never holds vacuously, for example.)
- Characterized inductively (by generation).

Admissibility: $J_1, \ldots, J_n \models J$.

- Is J provable under the assumptions that $J_1, \ldots, J_n$ are each provable?
- Is implication! May hold vacuously.
- Characterized coinductively (by behavior).

Can these be understood as function spaces?

Background

Derivability expresses syntactic structure of a logic:

- $\mathsf{tm} \vdash \mathsf{tm}$: a term with one free variable.
- $\phi\,\mathsf{true} \vdash \phi \wedge \phi\,\mathsf{true}$: a derivation with an assumption.

Admissibility expresses semantic structure of a logic:

- Side conditions on rules: $i \neq j$, ie $\neg(i = j)$, ie $i = j \models \bot$.
- Infinitary rules: $\phi(n)\,\mathsf{true}\ (\forall n \in \omega)$, ie $n \in \omega \models \phi(n)\,\mathsf{true}$.
- Meta-theorems: admissibility of cut, progress and preservation, etc.

We wish to intermix admissibility and derivability (contrast Twelf, Beluga, Delphin, etc).

Dependently Typed Syntax

Experience with LF says that dependent types are essential for representing syntax.

- Central notion: type-indexed families of types, eg $\phi\,\mathsf{true}$ indexed by $\phi : \mathsf{prop}$.
- Required for expressing metatheorems, eg admissibility of cut.

Admissibility is well-managed by the consequence relation of DTT:

$$d_1 : \phi_1\,\mathsf{true}, \ldots d_n : \phi_n\,\mathsf{true} \rhd D(d_1, \ldots, d_n) : \phi\,\mathsf{true}$$

Internalized as the ordinary (computational=negative) function space.

Dependently Typed Syntax

Derivability is managed using families of types indexed by contexts:

$$M : \mathsf{tm}[\Psi]$$

M represents a term with variables (parameters/indeterminates) $\Psi = u_1, \ldots, u_n$.

Can represent admissibilities between derivabilities, such as substitution:

$$x : \mathsf{tm}[\Psi], y : \mathsf{tm}[\Psi, u], \triangleright [x/u]y : \mathsf{tm}[\Psi].$$

Dependently Typed Syntax

The types $\mathsf{tm}[\Psi]$ "fit together" according to the structural properties of parameters:

- Permutation, contraction, weakening, substitution.
- Generally, if $\alpha : \Psi \leq \Psi'$, then $x : \mathsf{tm}[\Psi] \triangleright \mathsf{tm}[\alpha](x) : \mathsf{tm}[\Psi']$.

The family tm has a functorial action on the structure maps of contexts.

Want structural properties to emerge as generic programs, which are none other than the functorial actions of types!

Dependent Type Theory

Families of types respect equality of indices:

$$\frac{\Gamma \triangleright M : F[a] \quad \Gamma \triangleright a = b : A \quad \Gamma, x : A \triangleright F[x]\ \text{type}}{\Gamma \triangleright M : F[b]}$$

But what do we mean by equality?

Intensional: a and b are definitionally equivalent as a matter of their computational behavior (LF, Coq, etc).

Extensional: a and b are provably equivalent on the basis of their types (NuPRL).

May also consider pre-orders $a \leq b : A$ inducing subtyping $F[a] <: F[b]$.

Dependent Type Theory

ITT is minimal in the sense that it imposes the least requirements on families, and hence is compatible with various extensions. It generally retains decidability, but can be awkward to use in practice.

ETT is maximal in the sense that it imposes the strongest requirements on families, and hence is very easy to use. But it sacrifices decidability, and is not compatible with certain extensions and variations.

Enriched Type Theory

ETT is an instance of order-enriched type theory.

- Types are equipped with pre-orders or equivalence relations on their elements.
- Families must respect these orderings.
- cf order-enriched categories as they arise in domain theory.

But orderings are limited in their expressive power!

- Orderings have no computational significance: silently pass from $F[a]$ to $F[b]$.
- But there is, in general, no canonical way to impose an order or even an equivalence on a type!

Higher-Dimensional Type Theory

Example: universes.

- Want a type Set of sets and functions between them.
- What is an appropriate equality of Set's?
  - Mutual containment (standard).
  - Isomorphism of sets. But in which way?

There can be, in general, many isomorphisms (even automorphisms). eg, identity and swapping on Bool.

The evidence for equality is computationally significant (eg, functorial action of list on swapping map on Bool).

Higher-Dimensional Type Theory

Equip the equivalence relation on A with evidence $\alpha : a = b : A$.

- What is the form of α?
- How is α expoited?
- Introduced by Hofmann and Streicher.

A groupoid is an equivalence relation with evidence.

- Reflexivity: $\mathrm{id} : a = a : A$, a map from a to itself.
- Symmetry: $\alpha^{-1} : b = a : A$ if $\alpha : a = b : A$.
- Transitivity: $\beta \circ \alpha : a = c : A$ if $\alpha : a = b : A$ and $\beta : b = c : A$.

Generalize a pre-order to a category, require that all maps be invertible.

Higher-Dimensional Type Theory

Require that families respect the groupoid structure:

$$\frac{\alpha : a = b : A}{x : F[a] \triangleright \mathsf{map}(\alpha)(x) : F[b]}$$

Expresses the functorial action of F on α! That is, $\mathsf{map}(\alpha)$ is the generic program induced by α.

- Functors preserve inverses, so this is symmetric.
- Semantics of type theory relies heavily on symmetry!

Higher-Dimensional Type Theory

In ITT this structure is expressed propositionally using the identity type $I_A(a, b)$.

- Introduced by $\mathsf{refl}(a) : I_A(a, a)$.
- Eliminated by J, which reduces reasoning from $p : I_A(a, b)$ to reasoning from $\mathsf{refl}(a) : I_A(a, a)$.
- Symmetry and transitivity are derivable.
- Functorial action is definable.

Derivability of symmetry is problematic! For this reason I prefer the judgemental setup sketched here.

Higher-Dimensional Type Theory

What about the groupoid laws?

- id should be the unit of composition.
- Composition should be associative.
- Mutual inverses should compose to the identity.

In what sense do these laws hold?

- In no sense (so far), because "they don't have to."
- Strong sense: hold propositionally (when it makes sense).
- Weak sense: hold only up to higher equivalences.

This is where the higher-dimensional structure comes into play!

Higher-Dimensional Type Theory

To express the groupoid laws we require three-dimensional structure:

$$\gamma : \alpha = \beta : a = b : A$$

That is, γ is evidence that α and β are equivalent evidence that a and b are equal.

There needs to be enough evidence γ to ensure that this is a groupoid! But then we need groupoid structure on the γ's too!

Higher-Dimensional Type Theory

The general case is called a weak ω-groupoid; this provides an important point of contact with homotopy theory (in particular, the homotopy hypothesis).

Or we can cut off at whatever finite level we choose, and demand that the laws hold "on the nose". This amounts to imposing extensionality at higher dimensions!

For now we consider the two-dimensional case; it is an open problem to give a crisp presentation of the full higher-dimensional structure (cf ongoing work on ∞-categories).

Two-Dimensional Directed Type Theory

Example: contexts of parameters.

- Want a type Ctx of contexts, Ψ, described earlier.
- Enrich Ctx with evidence for the structural properties of parameters/variables/assumptions (ie, derivability).
- Use map to induce transformations on $\mathrm{tm}[\Psi]$: generic programming of structural properties.

The evidence for structurality is not symmetric!

- Must consider a category = pre-order with evidence.
- Cannot require invertibility of evidence.

Two-Dimensional Directed Type Theory

Equivalence relations are to groupoids as pre-orders are to categories.

- A category-enriched ("monoidoid") interpretation of types.
- Retain functorial action of transformations.

Transformations: $\alpha : a \leq b : A$.

- Closed under id and composition.
- Satisfy monoid laws on the nose (two-dimensional hypothesis).
- Induce action on families: $x : F[a] \triangleright \mathsf{map}(\alpha)(x) : F[b]$.

Two-Dimensional Directed Type Theory

The structure of type theory is radically altered!

- Must account for variances of types.
- Identity types $I_A(a, b)$ become Hom types, $H_A(a, b)$.

Both a contravariant (negative) and a covariant (positive) function space emerge naturally.

- Contravariant: computation/admissibility.
- Covariant: representation/derivability.
- cf Fiore/Tiuri/Plotkin; Washburn/Weirich/Chlipala.

Formulation of $H_A(a, b)$ unsolved: appears to require a modality for forming the opposite of a type.