Constructing Inductive Families in UniMath

Felix Rech
Advisor: Steven Schäfer

June 15, 2018

UniMath

- Dependent functions ($\prod_{a:A} B(a)$)
- Dependent pairs ($\sum_{a:A} B(a)$)
- Sum types ($A + B$)
- Equality ($a = b$)
- Universes ($\mathcal{U}_0, \mathcal{U}_1, \ldots$)
- Empty type, unit, bool and natural numbers
- Univalence
- Propositional resizing [Voevodsky 2011]

Not included

- No records
- No general inductive types
- No match construct

Mere Proposition

Definition

A type is a **mere proposition** if all inhabitants are equal.

Axiom (Propositional resizing)

Every mere proposition inhabits the smallest universe.

Propositional truncation

$\|A\|$ is a mere proposition and expresses that A is inhabited.

The Goal

General inductive types for UniMath

Side product:
Generic reasoning about inductive types

W-Types

```
Inductive W (A : Type) (B : A -> Type) :=
| sup : forall a : A, (B a -> W A B) -> W A B.
```

Example

$$\mathbb{N} \simeq \mathsf{W}(A, B) \text{ where}$$
$$A :\equiv \mathbf{2}$$
$$B :\equiv \lambda x, \textbf{if } x \textbf{ then } \mathbf{0} \textbf{ else } \mathbf{1}$$

W-Types

```
Inductive W (A : Type) (B : A -> Type) :=
| sup : forall a : A, (B a -> W A B) -> W A B.
```

Example

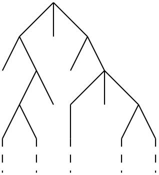
$$\mathbb{N} \simeq \mathsf{W}(A, B) \text{ where}$$
$$A :\equiv \mathbf{2}$$
$$B :\equiv \lambda x, \textbf{if } x \textbf{ then } \mathbf{0} \textbf{ else } \mathbf{1}$$

M-Types

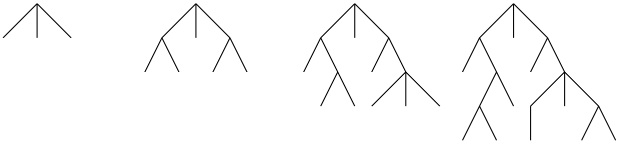
```
CoInductive W (A : Type) (B : A -> Type) :=
| sup : forall a : A, (B a -> W A B) -> W A B.
```

Construction of M-Types

Benedikt Ahrens, Paolo Capriotti, and Régis Spadotti.
"Non-wellfounded trees in homotopy type theory".
In: arXiv preprint arXiv:1504.02949 (2015)



Representation as sequence of approximations:

-  ...

Judgmental Computation Rule for M-Types

Given a coinductive type M with destructor dest and corecursor corec, we have a computation rule of the form

$$\mathsf{dest}\big(\mathsf{corec}(C, f, x)\big) = \phi(C, f, x)$$

for a certain ϕ.

We want this to hold by definition.

Judgmental Computation Rule for M-Types

Given a coinductive type M with destructor dest and corecursor corec, we have a computation rule of the form

$$\mathsf{dest}\big(\mathsf{corec}(C, f, x)\big) = \phi(C, f, x)$$

for a certain ϕ.

We want this to hold by definition.

The Solution: Remember C, f and x

$$\mathsf{M}' :\equiv \sum_{(m:\mathsf{M})} \sum_{(C,f,x)} \big(\mathsf{corec}(C, f, x) = m\big)$$

$$\begin{aligned} \mathsf{corec}'(C, f, x) &:\equiv \big(\mathsf{corec}(C, f, x), C, f, x, \mathsf{refl}\big) \\ \mathsf{dest}'\big((m, C, f, x)\big) &:\equiv \phi(C, f, x) \end{aligned}$$

Judgmental Computation Rule for M-Types

Given a coinductive type M with destructor dest and corecursor corec, we have a computation rule of the form

$$\mathsf{dest}\big(\mathsf{corec}(C,f,x)\big) = \phi(C,f,x)$$

for a certain ϕ.

We want this to hold by definition.

The Solution: Remember C, f and x

$$\mathsf{M}' :\equiv \sum_{(m:\mathsf{M})} \left\| \sum_{(C,f,x)} \big(\mathsf{corec}(C,f,x) = m\big) \right\|$$

$$\mathsf{corec}'(C,f,x) :\equiv \big(\mathsf{corec}(C,f,x), C, f, x, \mathsf{refl}\big)$$

We need to eliminate the truncation.

$$\mathsf{dest}'\big((m,C,f,x)\big) :\equiv \phi(C,f,x)$$

We need propositional resizing to use arbitrary C.

Construction of W-Types

$$\mathsf{W} :\equiv \sum_{m:\mathsf{M}} \| m \text{ satisfies the induction principle for } \mathsf{W} \|$$

Strictly Positive Types

Nested inductive and coinductive types with variables

$$A, B ::= K \mid x \mid A \times B \mid A + B \mid K \to A \mid \mu\, x.\, A \mid \nu\, x.\, A$$

where K is a constant type and x a variable.

Containers

[Abbott, Altenkirch, and Ghani 2005]

A polynomial-like normal form for functions from $\mathcal{U}$ to $\mathcal{U}$

Example (Lists)

$$\sum_{n:\mathbb{N}} \mathrm{Fin}(n) \to A$$

In General

$$\sum_{s:S} P(s) \to A$$

W-Types are the inductive fixed points of containers:

$$W(A,B) \simeq \sum_{a:A} B(a) \to W(A,B)$$

Construction of Strictly Positive Types

We generalize containers to describe functions from $(I \to \mathcal{U})$ to $\mathcal{U}$ for any I.

Theorem

Container functors are closed under all strictly positive type formers.

Inductive Families

```
Inductive Vec (A : Type) : nat -> Type :=
| vnil  : Vec A 0
| vcons : forall n, A -> Vec A n -> Vec A (S n).
```

$\mathrm{Vec}(A)$ is the inductive fixed point of a function from $(\mathbb{N} \to \mathcal{U})$ to $(\mathbb{N} \to \mathcal{U})$:

$$\begin{aligned} \mathrm{Vec}(A)_0 &\simeq \mathbf{1} \\ \mathrm{Vec}(A)_{n+1} &\simeq A \times \mathrm{Vec}(A)_n \end{aligned}$$

We need to generalize containers again for functions from $(I \to \mathcal{U})$ to $(J \to \mathcal{U})$ for any I and J.

Conclusion

What We Did

1. Construct indexed M-types from natural numbers
2. Construct indexed W-types from coinductive types
3. Obtain some computation rules by definition
4. Construct nested (co-)inductive families

Conclusion

What We Did

1. Construct indexed M-types from natural numbers
2. Construct indexed W-types from coinductive types
3. Obtain some computation rules by definition
4. Construct nested (co-)inductive families

Thank you!

References

Michael Abbott, Thorsten Altenkirch, and Neil Ghani. "Containers: constructing strictly positive types". In: *Theoretical Computer Science* 342.1 (2005), pp. 3–27.

Benedikt Ahrens, Paolo Capriotti, and Régis Spadotti. "Non-wellfounded trees in homotopy type theory". In: *arXiv preprint arXiv:1504.02949* (2015).

Vladimir Voevodsky. "Resizing rules, slides from a talk at TYPES2011". In: *At author's webpage* (2011).