Higher Inductive Types

Niels van der Weide, Henning Basold, Herman Geuvers

June 25, 2016

Our goal

- A syntax of higher inductive types.
- Definitional computation rules for points and paths.
- Semantical justification (restricted to nonrecursive HITs).

Related Work

- Lumsdaine and Shulman discuss semantics.
- Awodey and Sojakova give propositional computation rules.
- Van Doorn and Kraus describe how to obtain recursive higher inductive types from nonrecursive higher inductive types.
- Altenkirch, Capriotti, Dijkstra and Forsberg give a syntax, but no rules.

Intuition

- In HITs we additionally allow path constructors.
- For example, the circle is defined as

  `Inductive` $S^1 :=$
  $|$ base : S^1
  $|$ loop : base $=$ base

- Paths in X correspond with maps $I^1 \to X$.
- So, adding paths is adding images of maps $I^1 \to X$.
- For higher constructors: replace I^1 by I^n.

More concrete

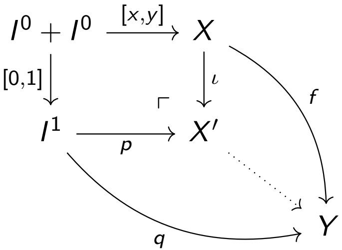
- Suppose, we have a type X with points x and y.
- To add $p : x = y$, we want to do a pushout

$$\begin{array}{ccc} I^0 + I^0 & \xrightarrow{[x,y]} & X \\ {\scriptstyle [0,1]}\downarrow & & \downarrow{\scriptstyle \iota} \\ I^1 & \xrightarrow[p]{} & X' \end{array}$$

- Note: UMP of pushout gives an elimination rule. The maps p and ι give the introduction rules.

Elimination Rule

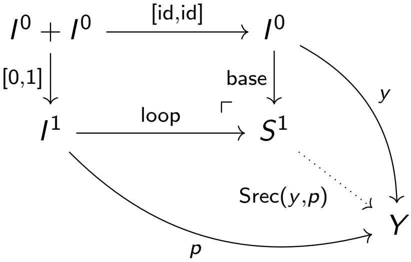
For the elimination rule we get



Given a path $q : I^1 \to Y$ and $f : X \to Y$ such that $q \circ [0,1] = f \circ [x,y]$, we get $X' \to Y$.

For S^1

We take $X = I^0$, and then $X' = S^1$.
Then a map $S^1 \to Y$ corresponds with a point $y : Y$ and a path $p : y = y$.



For S^1

So, for S^1 we get introduction rules:

$$\vdash \mathsf{base} : S^1, \quad \vdash \mathsf{loop} : \mathsf{base} = \mathsf{base}\,.$$

Furthermore, the elimination rule is

$$\frac{\vdash y : Y(\mathsf{base}) \qquad \vdash p : y =^{Y}_{\mathsf{loop}} y}{\vdash \mathsf{Srec}(y,p) : \prod x : S^1 . Y(x)}$$

And we have computation rules

$$\mathsf{Srec}(y,p)\ \mathsf{base} \equiv y, \quad \mathsf{apd}(\mathsf{Srec}(y,p), \mathsf{loop}) \equiv p.$$

Higher constructors

We can add higher paths in the same way

$\texttt{Inductive}\ I^{n+1} :=$
$\mid\ \text{top} : I^n \to I^{n+1}$
$\mid\ \text{bottom} : I^n \to I^{n+1}$
$\mid\ \text{middle} : \prod x : I^n.\, \text{top}\ x = \text{bottom}\ x$

For n-constructors of X we add images of maps $I^n \to X$.

Towards a General Definition

We first need some notation. Let T be a type and let $x_1 : A_1, \ldots, x_n : A_n$ be variables. We define $T(x_1, \ldots, x_n)$ to be the collection of terms t for which we can prove the judgment

$$x_1 : A_1, \ldots, x_n : A_n \vdash t : T.$$

General Definition

$\texttt{Inductive}\ T\ (B_1 : \textsc{Type}) \ldots (B_\ell : \textsc{Type}) :=$
$|\ \ c_1 : H_1(T) \to T$
$\ldots$
$|\ \ c_k : H_k(T) \to T$
$|\ \ p_1 : \prod x : A_1 . \overline{F_1} = \overline{G_1}$
$\ldots$
$|\ \ p_n : \prod x : A_n . \overline{F_n} = \overline{G_n}$

where

- Every H_i is polynomial
- A_i is any type depending on $B_1, \ldots B_\ell$
- $\overline{F_i}$ and $\overline{G_i}$ are terms in $(I^{d_i} \to T)(x, c_1, \ldots, c_k, p_1, \ldots, p_{i-1})$ with variables $x : A_i$, $c_j : H_j(T) \to T$ and $p_j : \overline{F_j} = \overline{G_j}$

Introduction Rules

We get introduction rules for the points

$$\frac{\vdash t : H_i(T)}{\vdash c_i\, t : T}$$

and the paths

$$\vdash p_i : \prod x : A_i . \overline{F_i} = \overline{G_i}.$$

Elimination Rule (nondependent)

Nondependent goes well.

$$\frac{\vdash z_i : H_i(Y) \to Y \text{ for } i = 1, \ldots, k \qquad \vdash q_i : \prod x : A_i.F'_i = G'_i \text{ for } i = 1, \ldots, n}{\vdash T\text{-elim}(z_1, \ldots, z_k, q_1, \ldots, q_n) : T \to Y}$$

where

$$F'_i = F_i[x, z_1, \ldots, z_k, q_1, \ldots, q_{i-1}]$$

$$G'_i = G_i[x, z_1, \ldots, z_k, q_1, \ldots, q_{i-1}].$$

Elimination Rule (dependent, work in progress)

Note: p_i gives an equality $p'_i : \overline{F_i}\, y = \overline{G_i}\, y$ for $y : I^{d_i}$. The elimination rule is

$$\frac{\begin{array}{c}\vdash z_i : \prod x : H_i(T).\overline{H_i}(Y)(x) \to Y(c_i\, x) \text{ for } i = 1, \ldots, k \\ \vdash q_i : \prod x : A_i. \prod y : I^{d_i}.F'_i\, y =^{Y}_{p'_i\, x} G'_i\, y \text{ for } i = 1, \ldots, n\end{array}}{\vdash T\text{-elim}(z_1, \ldots, z_k, q_1, \ldots, q_n) : \prod x : T.Y(x)}$$

where

$$F'_i = F_i[x, z_1, \ldots, z_k, q_1, \ldots, q_{i-1}]$$

$$G'_i = G_i[x, z_1, \ldots, z_k, q_1, \ldots, q_{i-1}].$$

Computation Rules (points)

We write $T\text{-elim}' = T\text{-elim}(z_1, \ldots, z_k, q_1, \ldots, q_n)$. We get a computation rule for the points

$$T\text{-elim}'\,(c_i\; t) \equiv (z_i\; t)\,(H_i(T\text{-elim}')\; t).$$

Computation Rules (paths), work in progress

For $f : \prod x : A.Y(x)$ and $p : I^n \to A$, We define $\mathrm{apd}(f, p)$ as $f \circ p : \prod x : I^n.Y(f\,(p\,x))$. So:

$$\mathrm{apd}(T\text{-elim'}, p_i\,t) : \prod x : I^{d_i+1}.Y(T\text{-elim'}\,(p_i\,t\,x))$$

Note: $q_i\,t$ gives a path

$$(q_i\,t)^* : \prod x : I^{d_i+1}.Y(T\text{-elim'}\,(p_i\,t\,x))$$

Then for all we say $t : A_i$

$$\mathrm{apd}(T\text{-elim}', p_i\,t) \equiv (q_i\,t)^*.$$

Examples

- Integers modulo m where m is fixed.

  `Inductive` $\mathbb{N}/m\mathbb{N} :=$
  $|\ 0 : \mathbb{N}/m\mathbb{N}$
  $|\ S : \mathbb{N}/m\mathbb{N} \to \mathbb{N}/m\mathbb{N}$
  $|\ \mathsf{mod} : S^m\, 0 = 0$

- Rational numbers. Here $\mathbb{Z}$ is the integers and $\mathbb{Z}_{\neq 0}$ is the nonzero integers.

  `Inductive` $\mathbb{Q} :=$
  $|\ \div : \mathbb{Z} \times \mathbb{Z}_{\neq 0} \to \mathbb{Q}$
  $|\ \mathtt{simplify} : \prod x : \mathbb{Z} \prod y : \mathbb{Z}_{\neq 0}.\ \frac{x}{y} = \frac{x \operatorname{div} \gcd(x,y)}{y \operatorname{div} \gcd(x,y)}$

Introduction Rules for $\mathbb{N}/m\mathbb{N}$

We have three introduction rules:

$$\vdash 0 : \mathbb{N}/m\mathbb{N},$$

$$\vdash S : \mathbb{N}/m\mathbb{N} \to \mathbb{N}/m\mathbb{N},$$

$$\vdash \mathsf{mod} : S^m\, 0 = 0.$$

Elimination Rule for $\mathbb{N}/m\mathbb{N}$

The elimination rule is

$$\frac{\vdash z : Y(0) \qquad \vdash s : \prod n : \mathbb{N}/m\mathbb{N}. Y(n) \to Y(S\,n) \qquad \vdash q : z =^{Y}_{\mathrm{mod}} s^n\, z}{\vdash \mathbb{N}/m\mathbb{N}\text{-elim}(z, s, q) : \prod x : \mathbb{N}/m\mathbb{N}. Y(x)}$$

Computation Rules for $\mathbb{N}/m\mathbb{N}$

The computation rules are

$$\mathbb{N}/m\mathbb{N}\text{-elim}(z, s, q)\, 0 \equiv z,$$

$$\mathbb{N}/m\mathbb{N}\text{-elim}(z, s, q)\, (S\; n) \equiv s\, (\mathbb{N}/m\mathbb{N}\text{-elim}(z, s, q)\; n),$$

$$\mathrm{apd}(\mathbb{N}/m\mathbb{N}\text{-elim}(z, s, q), \mathrm{mod}) \equiv q.$$

Another Example

- Consider

`Inductive` Cyl :=
| a : Cyl
| b : Cyl
| l : $a = a$
| r : $b = b$
| s : $\mathrm{lrec}(a, a, l) = \mathrm{lrec}(b, b, r)$

Note: we cannot give s with the type $l = r$. This shows the advantage of working with maps $I^n \to$ Cyl.

How to do the semantics?

- The paths are added via pushouts.
- We need interpretations of interval types I^n.
- Note: we also need to add paths like

$$\mathsf{ap}(S, \mathsf{mod}) : S\,0 = S\,(S^m\,0)$$

- But we need to guarantee that $\mathsf{ap}(S, \mathsf{refl}) = \mathsf{refl}$.

What about recursive HITs?

For recursive higher inductive types we do not have a justification of a syntax, but a proposal for a possible syntax.

$\texttt{Inductive}\ T\ (B_1 : \mathrm{Type}) \ldots (B_\ell : \mathrm{Type}) :=$
$|\ \ c_1 : H_1(T) \to T$
$\ldots$
$|\ \ c_k : H_k(T) \to T$
$|\ \ p_1 : \prod x : A_1(T). \overline{F_1} = \overline{G_1}$
$\ldots$
$|\ \ p_n : \prod x : A_n(T). \overline{F_n} = \overline{G_n}$

- H_i is polynomial.
- $\overline{F_i}$ and $\overline{G_i}$ are terms in $(I^{d_i} \to T)(x, c_1, \ldots, c_k, p_1, \ldots, p_{i-1})$ with variables $x : A_i(T)$, $c_j : H_j(T) \to T$ and $p_j : \overline{F_j} = \overline{G_j}$.
- $A_i(T)$ is any type depending on $B_1, \ldots B_\ell, T$, which is polynomial in T

Introduction Rules

The introduction rules for the points are

$$\frac{\vdash t : H_i(T)}{\vdash c_i\, t : T}$$

and the introduction rules for the paths are

$$\vdash p_i : \prod x : A_i(T).\overline{F_i} = \overline{G_i}$$

Elimination Rule

The elimination rule is

$$\frac{\begin{array}{c}\vdash z_i : H_i(Y) \to Y \quad \text{for } i = 1, \ldots, k \\ \vdash q_i : \prod x : A_i(Y).F_i' = G_i' \quad \text{for } i = 1, \ldots, n\end{array}}{\vdash T\text{-elim}(z_1, \ldots, z_k, q_1, \ldots, q_n) : T \to Y}$$

where

$$F_i' = F_i[x, z_1, \ldots, z_k, q_1, \ldots, q_{i-1}]$$

$$G_i' = G_i[x, z_1, \ldots, z_k, q_1, \ldots, q_{i-1}].$$

Computation Rules

We write $T\text{-elim}' = T\text{-elim}(z_1, \ldots, z_k, q_1, \ldots, q_n)$. The computation rules are for $t : H_i(T)$

$$T\text{-elim}'\,(c_i\;t) = z_i\,(H_i(T\text{-elim}')\;t)$$

and for all $t : A_i(T)$

$$\mathrm{apd}(T\text{-elim}', p_i\;t) = q_i\,(A_i(T\text{-elim}')\;t)$$

Examples

- Integers.

  $\texttt{Inductive}\ \mathbb{Z} :=$
  $|\ 0 : \mathbb{Z}$
  $|\ S : \mathbb{Z} \to \mathbb{Z}$
  $|\ P : \mathbb{Z} \to \mathbb{Z}$
  $|\ \texttt{inv}_1 : \prod x : \mathbb{Z}. S\,(P\,x) = x$
  $|\ \texttt{inv}_2 : \prod x : \mathbb{Z}. P\,(S\,x) = x$

  Something interesting: there are two different paths in $P\,(S\,(P\,0)) = P\,0$, so by Hedberg $\mathbb{Z}$ does not have decidable equality!

- Finite sets with elements from A as the free join-semilattice on A.

Conclusion

- Syntax for higher inductive types.
- Elimination rule and definitional computation rules.
- Semantics for nonrecursive HITs.

Further Work

- Extend semantics to recursive higher inductive types.
- Confluence and strong normalization of computation rules.
- Dependent HITs.
- Version in Cubical Type Theory.