# The Three-HITs Theorem

Andrej Bauer, Niels van der Weide

April 26, 2017

*"A canonical type A is defined by prescribing how a canonical object of type A is formed as well as how two equal canonical objects of type A are formed. There is no limitation on this prescription except that the relation of equality which it defines between canonical objects of type A must be reflexive, symmetric and transitive. If the rules for forming canonical objects as well as equal canonical objects of a certain type are called the introduction rules for that type, we may thus say with Gentzen(1934) that a canonical type (proposition) is defined by its introduction rules."*

“*A canonical type A is defined by prescribing how a canonical object of type A is formed as well as how two equal canonical objects of type A are formed. There is no limitation on this prescription except that the relation of equality which it defines between canonical objects of type A must be reflexive, symmetric and transitive. If the rules for forming canonical objects as well as equal canonical objects of a certain type are called the introduction rules for that type, we may thus say with Gentzen(1934) that a canonical type (proposition) is defined by its introduction rules.*”
Martin-Löf, Per. "Constructive mathematics and computer programming." *Studies in Logic and the Foundations of Mathematics* 104 (1982): 153-175.

"*A canonical type A is defined by prescribing how a canonical object of type A is formed as well as how two equal canonical objects of type A are formed. There is no limitation on this prescription except that the relation of equality which it defines between canonical objects of type A must be reflexive, symmetric and transitive. If the rules for forming canonical objects as well as equal canonical objects of a certain type are called the introduction rules for that type, we may thus say with Gentzen(1934) that a canonical type (proposition) is defined by its introduction rules.*" Martin-Löf, Per. "Constructive mathematics and computer programming." *Studies in Logic and the Foundations of Mathematics* 104 (1982): 153-175.

## Higher Inductive Types

Higher inductive type (HIT): generated by inductive point constructors and path constructors.
Canonical types in Martin-Löf's sense corresponds with higher inductive types in HoTT.

## Higher Inductive Types

However, how can HITs be constructed?

## Constructing Inductive Types

An inductive type $T$ with a constructor $c : F\ T \to T$ is constructed as a colimit.

$$\mathbf{0} \to F\,\mathbf{0} \to F(F\,\mathbf{0}) \to \dots$$

Idea: same for higher inductive types, but make identifications on the way.

## The Three-HITs Theorem

Theorem: all higher inductive types can be constructed from three specific HITs.
These HITs represent the colimit and making identifications.

This is work in progress. More details and the Coq formalization can be found on.
`https://github.com/nmvdw/Three-HITs`

## Syntax of HITs

For a higher inductive type, we want to add equations like

$$\prod x : A, t = r$$

With $t$ and $r$ ‘canonical terms’.

## Syntax of HITs

For a higher inductive type, we want to add equations like

$$\prod x : A, t = r$$

With $t$ and $r$ 'canonical terms'.
This means the scheme looks something like

Inductive $T\ (B_1 : \textsc{Type}) \ldots (B_\ell : \textsc{Type}) :=$
$|\ \ c_1 : H_1[T\ B_1 \cdots B_\ell] \to T\ B_1 \cdots B_\ell$
$\ldots$
$|\ \ c_k : H_k[T\ B_1 \cdots B_\ell] \to T\ B_1 \cdots B_\ell$
$|\ \ p_1 : \prod(x : A_1[T\ B_1 \cdots B_\ell]), t_1 = r_1$
$\ldots$
$|\ \ p_n : \prod(x : A_n[T\ B_1 \cdots B_\ell]), t_n = r_n$

## Constructor Terms

We start with:

- We have context $\Gamma$;
- We have $c_i : H_i(T) \to T$ (given by inductive type);
- We have a parameter $x : A[T]$ with $A$ polynomial functor.

## Building Constructor Terms

$$\frac{\Gamma \vdash t : B \qquad T \text{ does not occur in } B}{x : A \Vdash t : B} \qquad \frac{}{x : A \Vdash x : A}$$

## Building Constructor Terms

$$\frac{\Gamma \vdash t : B \qquad T \text{ does not occur in } B}{x : A \Vdash t : B} \qquad \frac{}{x : A \Vdash x : A}$$

$$\frac{j \in \{1, 2\} \qquad x : A \Vdash r : G_1 \times G_2}{x : A \Vdash \pi_j\, r : G_j}$$

$$\frac{j = \{1, 2\} \qquad x : A \Vdash r_j : G_j}{x : A \Vdash (r_1, r_2) : G_1 \times G_2}$$

## Building Constructor Terms

$$\frac{\Gamma \vdash t : B \qquad T \text{ does not occur in } B}{x : A \Vdash t : B} \qquad \frac{}{x : A \Vdash x : A}$$

$$\frac{j \in \{1,2\} \qquad x : A \Vdash r : G_1 \times G_2}{x : A \Vdash \pi_j\, r : G_j}$$

$$\frac{j = \{1,2\} \qquad x : A \Vdash r_j : G_j}{x : A \Vdash (r_1, r_2) : G_1 \times G_2}$$

$$\frac{j \in \{1,2\} \qquad x : A \Vdash r : G_j}{x : A \Vdash \mathrm{in}_j\, r : G_1 + G_2}$$

## Building Constructor Terms

$$\frac{\Gamma \vdash t : B \qquad T \text{ does not occur in } B}{x : A \Vdash\ t : B} \qquad \frac{}{x : A \Vdash\ x : A}$$

$$\frac{j \in \{1,2\} \qquad x : A \Vdash\ r : G_1 \times G_2}{x : A \Vdash\ \pi_j\, r : G_j}$$

$$\frac{j = \{1,2\} \qquad x : A \Vdash\ r_j : G_j}{x : A \Vdash\ (r_1, r_2) : G_1 \times G_2}$$

$$\frac{j \in \{1,2\} \qquad x : A \Vdash\ r : G_j}{x : A \Vdash\ \mathrm{in}_j\, r : G_1 + G_2}$$

$$\frac{x : A \Vdash\ r : H_i[T]}{x : A \Vdash\ c_i\, r : T}$$

## The Scheme

$$
\begin{array}{l}
\texttt{Inductive}\ T\ (B_1 : \mathrm{Type}) \ldots (B_\ell : \mathrm{Type}) := \\
\mid\ c_1 : H_1[T\ B_1 \cdots B_\ell] \to T\ B_1 \cdots B_\ell \\
\ldots \\
\mid\ c_k : H_k[T\ B_1 \cdots B_\ell] \to T\ B_1 \cdots B_\ell \\
\mid\ p_1 : \prod(x : A_1[T\ B_1 \cdots B_\ell]), t_1 = r_1 \\
\ldots \\
\mid\ p_n : \prod(x : A_n[T\ B_1 \cdots B_\ell]), t_n = r_n
\end{array}
$$

Here we have

- $H_i$ and $A_j$ are polynomials;
- $t_j$ and $r_j$ are constructor terms over $c_1, \ldots, c_k$ with $x : A_j \Vdash t_j, r_j : T$.

Note: all HITs in this talk are finitary. Also, only 1-HITs.

## Introduction Rules

$$\frac{\Gamma \vdash B_1 : \textsc{Type} \qquad \cdots \qquad \Gamma \vdash B_\ell : \textsc{Type}}{\Gamma \vdash T\ B_1 \cdots B_\ell : \textsc{Type}}$$

$$\frac{\vdash \Gamma \quad \textsc{Ctx}}{\Gamma \vdash c_i : H_i[T] \to T}$$

$$\frac{\vdash \Gamma \quad \textsc{Ctx}}{\Gamma \vdash p_j : \prod(x : A_j[T]) \to t_j = r_j}$$

## Lifting Constructor Terms

To lift a constructor term $x : A[T] \Vdash r : G[T]$, we need:

- Constructors $c_i : H_i[X] \to X$;
- A type family $Y : T \to \textsc{Type}$;
- Terms $\Gamma \vdash f_i : \prod(x : H_i[T]), \overline{H}_i(Y)\, x \to Y(c_i\, x)$.

## Lifting Constructor Terms

To lift a constructor term $x : A[T] \Vdash r : G[T]$, we need:

- Constructors $c_i \colon H_i[X] \to X$;
- A type family $Y \colon T \to \textsc{Type}$;
- Terms $\Gamma \vdash f_i : \prod(x : H_i[T]), \overline{H}_i(Y)\, x \to Y(c_i\, x)$.

Then we define

$$\Gamma, x : A[T], h_x : \overline{A}(Y)\, x \vdash \widehat{r} : \overline{G}(Y)\, r$$

## Lifting Constructor Terms

To lift a constructor term $x : A[T] \Vdash r : G[T]$, we need:

- Constructors $c_i \colon H_i[X] \to X$;
- A type family $Y \colon T \to \textsc{Type}$;
- Terms $\Gamma \vdash f_i : \prod(x : H_i[T]), \overline{H}_i(Y)\, x \to Y(c_i\, x)$.

Then we define

$$\Gamma, x : A[T], h_x : \overline{A}(Y)\, x \vdash \widehat{r} : \overline{G}(Y)\, r$$

by induction as follows

$$\widehat{t} := t \qquad \widehat{x} := h_x \qquad \widehat{c_i\, r} := f_i\; r\; \widehat{r}$$
$$\widehat{\pi_j\, r} := \pi_j\, \widehat{r} \qquad \widehat{(r_1, r_2)} := (\widehat{r_1}, \widehat{r_2}) \qquad \widehat{\mathrm{in}_j\, r} := \widehat{r}$$

## Elimination Rule

$$\frac{\begin{array}{c} Y \colon T \to \textsc{Type} \\ \Gamma \vdash f_i : \prod(x : H_i[T]), \overline{H}_i(Y)\, x \to Y\,(c_i\, x) \\ \Gamma \vdash q_j : \prod(x : A_j[T])(h_x : \overline{A}_j(Y)\, x), \widehat{t_j} =^{Y}_{(p_j\, x)} \widehat{r_j} \end{array}}{\Gamma \vdash T\,\mathrm{rec}(f_1, \ldots, f_k, q_1, \ldots, q_n) : \prod(x : T), Y\, x}$$

Note that $\widehat{t_j}$ and $\widehat{r_j}$ depend on all the $f_i$.

## Elimination Rule

$$\frac{\begin{array}{c} Y \colon T \to \mathrm{Type} \\ \Gamma \vdash f_i : \prod(x : H_i[T]), \overline{H}_i(Y)\, x \to Y\,(c_i\, x) \\ \Gamma \vdash q_j : \prod(x : A_j[T])(h_x : \overline{A}_j(Y)\, x), \widehat{t_j} =^{Y}_{(p_j\, x)} \widehat{r_j} \end{array}}{\Gamma \vdash T\,\mathrm{rec}(f_1, \ldots, f_k, q_1, \ldots, q_n) : \prod(x : T), Y\, x}$$

Note that $\widehat{t_j}$ and $\widehat{r_j}$ depend on all the $f_i$.

## Computation Rules

$$T\,\mathrm{rec}\,(c_i\ t) = f_i\ t\ (\overline{H}_i(T\,\mathrm{rec})\ t),$$
$$\mathrm{apD}\ T\,\mathrm{rec}\ p_j\ a = q_j\ a\ (\overline{A}_j(T\,\mathrm{rec})\ a).$$

## Which HITs do we need?

Remember: we will construct HITs in a similar way as inductive types, but with identifications along the way.
We need HITs for

- Making identifications
- Colimits

## Which HITs do we need?

Points are identified via the coequalizer.

```
Inductive coeq $(A, B : \textsc{Type})$ $(f, g : A \to B)$ $:=$
| inC : $B \to$ coeq $A\ B\ f\ g$
| glueC : $\prod(a : A),$ inC $(f\ a) =$ inC $(g\ a)$
```

## Which HITs do we need?

Points are identified via the coequalizer.

$\texttt{Inductive}$ coeq $(A, B : \mathrm{T{\small YPE}})\ (f, g : A \to B) :=$
$|\ \ \mathsf{inC} : B \to \mathsf{coeq}\ A\ B\ f\ g$
$|\ \ \mathsf{glueC} : \prod(a : A), \mathsf{inC}\ (f\ a) = \mathsf{inC}\ (g\ a)$

Elimination rule.

$$\frac{\vdash Y : \mathsf{coeq}\ A\ B\ f\ g \to \mathrm{T{\small YPE}} \qquad \vdash i_Y : \prod(b : B), Y\ (\mathsf{inC}\ b) \qquad \vdash g_Y : \prod(a : A), \mathsf{glueC}_*(i_Y\ (f\ a)) = i_Y\ (g\ a)}{\vdash \mathsf{coeqind}(i_Y, g_Y) : \prod(x : \mathsf{coeq}\ A\ B\ f\ g), Y\ x}$$

## Which HITs do we need?

Colimits.

`Inductive` colim $(F : \mathbb{N} \to \textsc{Type})\ (f : \prod(n : \mathbb{N}, F\ n \to F(n+1))) :=$
$|$ inc : $\prod(n : \mathbb{N}), F\ n \to$ colim $F\ f$
$|$ com : $\prod(n : \mathbb{N})(x : F\ n), \mathsf{inc}\ n\ x = \mathsf{inc}\ (n+1)\ (f\ n\ x)$

## Which HITs do we need?

Colimits.

$\texttt{Inductive}$ colim $(F : \mathbb{N} \to \textsc{Type})\ (f : \prod(n : \mathbb{N}, F\ n \to F(n+1))) :=$
$|\ \ \mathsf{inc} : \prod(n : \mathbb{N}), F\ n \to \mathsf{colim}\ F\ f$
$|\ \ \mathsf{com} : \prod(n : \mathbb{N})(x : F\ n), \mathsf{inc}\ n\ x = \mathsf{inc}\ (n+1)\ (f\ n\ x)$

Elimination rule:

$$\frac{\begin{array}{c}\vdash Y : \mathsf{colim}\ F\ f \to \textsc{Type} \\ \vdash i_Y : \prod(n : \mathbb{N})(x : F\ n), Y\ (\mathsf{inc}\ n\ x) \\ \vdash c_Y : \prod(n : \mathbb{N})(x : F\ n), \mathsf{com}_*(i_Y\ n\ x) = i_Y\ (n+1)\ (f\ n\ x)\end{array}}{\vdash \mathsf{colimind}(i_Y, c_Y) : \prod(x : \mathsf{colim}\ F\ f), Y\ x}$$

## Which HITs do we need?

We also need to identify paths.
Start with a type $B$, and suppose we have a family of paths with the same endpoints.

$$p : A \to \sum b_1, b_2 : B, (b_1 = b_2) \times (b_1 = b_2).$$

Write $p_i$ for the $i$th coordinate of $p$. We want to identify $p_3\ a$ and $p_4\ a$ for $a \in A$.

## Which HITs do we need?

We also need to identify paths.
Start with a type $B$, and suppose we have a family of paths with the same endpoints.

$$p : A \to \sum b_1, b_2 : B, (b_1 = b_2) \times (b_1 = b_2).$$

Write $p_i$ for the $i$th coordinate of $p$. We want to identify $p_3\ a$ and $p_4\ a$ for $a \in A$.

```
Inductive pcoeq
(A, B : Type)
(p : A → ∑(b₁, b₂ : B), (b₁ = b₂) × (b₁ = b₂)) :=
| inP : B → pcoeq A B f g
| glueP : ∏(a : A), ap inP (p₃ a) = ap inP (p₄ a)))
```

Note: we need ap in the path expressions.
See Dybjer, Moeneclay.

## Which HITs do we need?

We also need to identify paths.
Start with a type $B$, and suppose we have a family of paths with the same endpoints.

$$p : A \to \sum b_1, b_2 : B, (b_1 = b_2) \times (b_1 = b_2).$$

Write $p_i$ for the $i$th coordinate of $p$. We want to identify $p_3\ a$ and $p_4\ a$ for $a \in A$.

```
Inductive pcoeq
(A, B : Type)
(p : A → ∑(b₁, b₂ : B), (b₁ = b₂) × (b₁ = b₂)) :=
| inP : B → pcoeq A B f g
| glueP : ∏(a : A), ap inP (p₃ a) = ap inP (p₄ a)))
```

Note: we need ap in the path expressions.
See Dybjer, Moeneclay.

## Elimination Rule of pcoeq (Naive Attempt)

Replacing ap by apD does not work.

$$\frac{\begin{array}{c}\vdash Y : \mathsf{pcoeq}\ A\ B\ f\ g \to \textrm{Type} \\ \vdash i_Y : \prod(b : B), Y\ (\mathsf{inP}\ b) \\ \vdash g_Y : \prod(a : A), \mathsf{apD}\ \ i_Y\ (p_3\ a) = \mathsf{apD}\ \ i_Y\ (p_4\ a)\end{array}}{\vdash \mathsf{pcoeqind}(i_Y, g_Y) : \prod(x : \mathsf{pcoeq}\ A\ B\ f\ g), Y\ x}$$

Note that this is not well-typed.

$$\mathsf{apD}\ \ i_Y\ (p_3\ a) : (p_3)_*(i_Y\ b_1) = i_Y\ b_2,$$

$$\mathsf{apD}\ \ i_Y\ (p_4\ a) : (p_4)_*(i_Y\ b_1) = i_Y\ b_2.$$

## Elimination Rule of pcoeq (Correct Attempt)

But we can relate them.

**Lemma**

*Given is* $Y : \mathsf{pcoeq} \to \textsc{Type}$ *and* $i_Y : \prod(b : B), Y(\mathsf{inP}\ b)$. *Then we have a term*

$$\mathsf{coh} : \prod(a : A), (p_3\ a)_*(i_Y\ b_1) = (p_4\ a)_*(i_Y\ b_1)$$

## Elimination Rule of pcoeq (Correct Attempt)

But we can relate them.

Lemma

*Given is* $Y : \mathsf{pcoeq} \to \mathrm{TYPE}$ *and* $i_Y : \prod(b : B), Y(\mathsf{inP}\ b)$*. Then we have a term*

$$\mathsf{coh} : \prod(a : A), (p_3\ a)_*(i_Y\ b_1) = (p_4\ a)_*(i_Y\ b_1)$$

Not difficult, but we need a term of type

$$\prod(P : B \to Type) \prod(f : A \to B) \prod(p : x = y) \prod(z : P(f\ x)),$$
$$p_*^{\lambda a, P(f\ a)} z = (ap\ f\ p)_*^P\ z$$

This follows by path induction.

## Elimination Rule of pcoeq (Correct Attempt)

With the coherency we can give the right elimination rule.

$$\frac{\begin{array}{c}\vdash Y : \mathrm{pcoeq}\ A\ B\ f\ g \to \textsc{Type} \\ \vdash i_Y : \prod(b : B), Y\ (\mathrm{inP}\ b) \\ \vdash g_Y : \prod(a : A), (\mathrm{coh}\ \ a)^{-1} \bullet \mathrm{apD}\ \ i_Y\ (p_3\ a) = \mathrm{apD}\ \ i_Y\ (p_4\ a)\end{array}}{\vdash \mathrm{pcoeqind}(i_Y, g_Y) : \prod(x : \mathrm{pcoeq}\ A\ B\ f\ g), Y\ x}$$

## Elimination Rule of pcoeq (Correct Attempt)

With the coherency we can give the right elimination rule.

$$\frac{\begin{array}{c}\vdash Y : \mathsf{pcoeq}\ A\ B\ f\ g \to \textsc{Type} \\ \vdash i_Y : \prod(b : B), Y\ (\mathsf{inP}\ b) \\ \vdash g_Y : \prod(a : A), (\mathsf{coh}\ \ a)^{-1} \bullet \mathsf{apD}\ \ i_Y\ (p_3\ a) = \mathsf{apD}\ \ i_Y\ (p_4\ a)\end{array}}{\vdash \mathsf{pcoeqind}(i_Y, g_Y) : \prod(x : \mathsf{pcoeq}\ A\ B\ f\ g), Y\ x}$$

Note:

$$\begin{aligned}
\mathsf{apD}\ \ i_Y\ (p_3\ a) &: (p_3)_*(i_Y\ b_1) = i_Y\ b_2 \\
\mathsf{coh}\ \ a &: (p_3\ a)_*(i_Y\ b_1) = (p_4\ a)_*(i_Y\ b_1) \\
(\mathsf{coh}\ \ a)^{-1} &: (p_4\ a)_*(i_Y\ b_1) = (p_3\ a)_*(i_Y\ b_1) \\
(\mathsf{coh}\ \ a)^{-1} \bullet \mathsf{apD}\ \ i_Y\ (p_3\ a) &: (p_4)_*(i_Y\ b_1) = i_Y\ b_2 \\
\mathsf{apD}\ \ i_Y\ (p_4\ a) &: (p_4)_*(i_Y\ b_1) = i_Y\ b_2
\end{aligned}$$

## The Three-HITs Theorem

**Theorem (Three-HITs Theorem)**

*In Martin-Löf type theory extended with a coequalizers, path coequalizers and homotopy colimits, we can interpret each higher inductive type. This means that for each HIT we can define a type with the same introduction, elimination and computation rules.*

## The Three-HITs Theorem

In Coq:

- Extend the language with coequalizers, colimits and path coequalizers (using axioms).
- Define signatures of HITs. This represents the given syntax of HITs.
- A HIT on a signature is a type with interpretations of the introduction, elimination and computation rules.
- Then the Three-HITs Theorem says: each signature has a HIT.

## Idea of the Proof

The constructions in the proof are complicated. We will demonstrate it with examples.

## Idea of the Proof

The constructions in the proof are complicated. We will demonstrate it with examples.
For general $H$ it works as follows.

- Build sequence of approximations $F : \mathbb{N} \to \textsc{Type}$.
- Add point constructors at every step. For $c : A\,H \to H$ we look at $F\,n + A(F\,n)$.
- Make identifications for path constructors whenever possible.
- Identify duplicate points or paths.
- The map $F\,n \to F(n+1)$ is composition of inclusions and quotient maps.

## Idea of the Proof: Obstacles

Main obstacles: recursion.

- A constructor $c : T \to T$ and a path $p : t = r$ also gives paths $\mathrm{ap}\ c\ p$.
- For the truncation

  $\texttt{Inductive}\ ||*|| :=$
  $|\ \ a : \mathbf{1} \to ||*||$
  $|\ \ p : \prod(x, y : ||*||)), x = y$

  We have $p\,(a *)\,(a *) : a * = a *$

## Idea of the Proof: Obstacles

Main obstacles: recursion.

- A constructor $c : T \to T$ and a path $p : t = r$ also gives paths ap $c$ $p$.
- For the truncation

  ```
  Inductive $|| * || :=$
  | $a : \mathbf{1} \to || * ||$
  | $p : \prod(x, y : || * ||)), x = y$
  ```

  We have $p\,(a\,*)\,(a\,*) : a\,* = a\,*$, and a path $s : p\,(a\,*)\,(a\,*) = \text{refl}$.

## Simple Example

No recursion.

$\texttt{Inductive}\ I^1 :=$
$|\ \ c : A\, I^1 \to I^1$
$|\ \ s : c\, 0 = c\, 1$

Define $A\, X = \mathbf{2}$. Call the points $z$ and $o$.
Construct $F\, 0$ as follows.

- Start with $A\, 0 = \mathbf{2}$.
- We can make the identifications. This gives the interval.

We continue to $F\,\mathbf{1}$.

- Start with $I^1 + \mathbf{2}$.
- We can make the identification: we identify $z$ and $o$ in the second component.
- We have $I^1 + I^1$ now.
- Two copies of $z$ and $o$; identify them with coequalizer.
- Two copies of $s$: identify them with path coequalizer.
- This results in $I^1$.

## Idea of the Proof: Recursive Points

Now $\mathbb{N}_1$: natural numbers modulo 1.

```
Inductive $\mathbb{N}_1$ :=
| $0 : \mathbb{N}_1$
| $S : \mathbb{N}_1 \to \mathbb{N}_1$
| $m : 0 = S\,0$.
```

We also need the paths ap $S$ $m$.

## Idea of the Proof: Recursive Points

Start with $F\,\mathbf{0}$.

- We add a point 0.
- No identifications can be made.

## Idea of the Proof: Recursive Points

For $F\,\mathbf{1}$:

- Add points $0'$ and $S\,0$.
- Identify $0$ and $S\,0$ (path constructor).
- Identify $0$ and $0'$ (duplicates).

## Idea of the Proof: Recursive Points

In $F\,\mathbf{2}$ we want to find ap $S\ m$.

- We start with $F\,\mathbf{1} + A(F\,\mathbf{1}) = F\,\mathbf{1} + (\mathbf{1} + F\,\mathbf{1})$.
- Successor of $x : F\,\mathbf{1}$ is $\mathsf{in}_3\ x$ in $f\,\mathbf{2}$.
- Then ap $S\ m$ is ap $\mathsf{in}_3\ m$ in $F\,\mathbf{2}$.

## Idea of the Proof: Recursive Paths

Truncation of the point.

`Inductive` $|| * || :=$
$|\;\; a : \mathbf{1} \to || * ||$
$|\;\; p : \prod(x, y : || * ||)), x = y$

## Idea of the Proof: Recursive Paths

Start with $F\,\mathbf{0}$.

- We add a point $a$.
- Add a path $p\,a\,a : a = a$.
- No duplicates.

This gives $S^1$.

## Idea of the Proof: Recursive Paths

The interesting thing happens at $F\,\mathbf{1}$.

- We add a point $a'$.
- Add a path $p\;a'\;a' : a' = a'$.
- We identify $a$ and $a'$ and $p\;a\;a$ and $p\;a'\;a'$.

## Idea of the Proof: Recursive Paths

The interesting thing happens at $F\,\mathbf{1}$.

- We add a point $a'$.
- Add a path $p\,a'\,a' : a' = a'$.
- We identify $a$ and $a'$ and $p\,a\,a$ and $p\,a'\,a'$.
- But more happens.

## Idea of the Proof: Recursive Paths

- Let's focus on the first $S^1$.
- For $x, y : S^1$ we add a path $p\,x\,y : x = y$.
- This is *not* $S^1$.

## Idea of the Proof: Recursive Paths

Basically, the following construction happens.

Inductive $|A|$ $(A : \mathrm{T{\small YPE}}) :=$
$|$ $a : A \to |A|$
$|$ $p : \prod(x, y : A), a\,x = a\,y$

$$|A|, |\,|A|\,|, \ldots$$

## Conclusion

- Finitary HITs can be constructed from three simple HITs.
- The construction is done in type theory.
- Disadvantage: the acquired computation rules are propositional equalities.
- More details and Coq code can be found on: `https://github.com/nmvdw/Three-HITs`