Programming with Higher Inductive Types

Henning Basold, Herman Geuvers, Niels van der Weide

January 31, 2017

How to define Finite Sets

- Represent a set as a list of elements.
- Operations on sets then become operations on lists.
- However, then our implementation needs to maintain several invariants.

How to define Finite Sets according to Kuratowski

A more logical definition would be

$\texttt{Inductive}\ \mathrm{Fin}(\_)\ (A : \mathit{Type}) :=$
$|\ \emptyset : \mathrm{Fin}(A)$
$|\ L : A \to \mathrm{Fin}(A)$
$|\ \cup : \mathrm{Fin}(A) \times \mathrm{Fin}(A) \to \mathrm{Fin}(A)$

and we require some equations (eg: $\cup$ is commutative, associative, $\emptyset$ is neutral, ...).

How to define Finite Sets according to Kuratowski

A more logical definition would be

$\texttt{Inductive}\ \mathrm{Fin}(\_)\ (A : \mathit{Type}) :=$
$|\ \emptyset : \mathrm{Fin}(A)$
$|\ L : A \to \mathrm{Fin}(A)$
$|\ \cup : \mathrm{Fin}(A) \times \mathrm{Fin}(A) \to \mathrm{Fin}(A)$

and we require some equations (eg: $\cup$ is commutative, associative, $\emptyset$ is neutral, . . .).
However, inductive types are 'freely generated'. We can't allow extra equations.

Possible solutions

1. Data Types with laws
2. Quotient Types
3. Quotient Inductive-Inductive Types
4. Higher Inductive Types

We will look at the last solution.

- Published as ‘Higher Inductive Types in Programming’.
- Formalized in Coq using the HoTT library by Bauer, Gross, Lumsdaine, Shulman, Sozeau, Spitters.

Approach

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

$$\prod x : A, f\, x = g\, x$$

Approach

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

$$\prod x : A, f\, x = g\, x$$

This means the scheme looks something like

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

Approach

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

$$\prod x : A, f\,x = g\,x$$

This means the scheme looks something like

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

However, for arbitrary A_i, t_i, r_i deducing the elimination rule is difficult.

Constructor Terms

We start with:

- We have context Γ;
- 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

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

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 $x : A_j \Vdash t_j, r_j : T$.

Example: Finite Sets

```
Inductive Fin(_) (A : Type) :=
| ∅ : Fin(A)
| L : A → Fin(A)
| ∪ : Fin(A) × Fin(A) → Fin(A)
| as : ∏(x, y, z : Fin(A)), ∪(x, ∪(y, z)) = ∪(∪(x, y), z)
| neut₁ : ∏(x : Fin(A)), ∪(x, ∅) = x
| neut₂ : ∏(x : Fin(A)), ∪(∅, x) = x
| com : ∏(x, y : Fin(A)), ∪(x, y) = ∪(y, x)
| idem : ∏(x : A), ∪(L x, L x) = L x
```

Example: Finite Sets

```
Inductive Fin(_) (A : Type) :=
| ∅ : Fin(A)
| L : A → Fin(A)
| ∪ : Fin(A) × Fin(A) → Fin(A)
| as : ∏(x, y, z : Fin(A)), ∪(x, ∪(y, z)) = ∪(∪(x, y), z)
| neut₁ : ∏(x : Fin(A)), ∪(x, ∅) = x
| neut₂ : ∏(x : Fin(A)), ∪(∅, x) = x
| com : ∏(x, y : Fin(A)), ∪(x, y) = ∪(y, x)
| idem : ∏(x : A), ∪(L x, L x) = L x
```

Note:

$$x : A \Vdash x : A \qquad x : \mathsf{Fin}(A) \Vdash x : \mathsf{Fin}(A)$$

$$x : A \Vdash L\,x : \mathsf{Fin}(A) \qquad x : \mathsf{Fin}(A) \Vdash \emptyset : \mathsf{Fin}(A)$$

$$x : A \Vdash \cup(L\,x, L\,x) : \mathsf{Fin}(A) \qquad x : \mathsf{Fin}(A) \Vdash (x, \emptyset) : \mathsf{Fin}(A)$$

$$x : \mathsf{Fin}(A) \Vdash \cup(x, \emptyset) : \mathsf{Fin}(A)$$

Introduction Rules

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

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

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

Lifting Constructor Terms

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

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

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 $U : T \to \mathit{Type}$;
- Terms $\Gamma \vdash f_i : (x : H_i[T]) \to \overline{H}_i(U)\, x \to U(c_i\, x)$.

Then we define

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

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 $U \colon T \to \mathit{Type}$;
- Terms $\Gamma \vdash f_i : (x : H_i[T]) \to \overline{H}_i(U)\,x \to U(c_i\,x)$.

Then we define

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

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

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

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

Elimination Rule

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

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

Computation Rules

$$T\text{-rec}\,(c_i\ t) \longrightarrow f_i\ t\,(\overline{H}_i(T\text{-rec})\ t),$$
$$\text{apd}(T\text{-rec}, p_j\ a) \longrightarrow q_j\ a\,(\overline{A}_j(T\text{-rec})\ a).$$

Elimination Rule for Kuratowski Sets

$$\dfrac{\begin{array}{c} Y : \ \mathrm{Fin}(A) \to \mathit{Type} \\ \emptyset_Y : Y[\emptyset] \\ L_Y : \prod(a : A), Y[L\, a] \\ \cup_Y : \prod(x, y : \mathrm{Fin}\, A), Y[x] \times Y[y] \to Y[\cup(x, y)] \\ a_Y : \prod(x, y, z : \mathrm{Fin}(A)) \prod(a : Y[x]) \prod(b : Y[y]) \prod(c : Y[z]), \\ \cup_Y\, x\, (\cup(y, z))\, (a, (\cup_Y\, y\, z\, (b, c))) =^Y_{\mathrm{as}} \cup_Y\, (\cup(x, y))\, z\, ((\cup_Y\, x\, y\, (a, b)), c) \\ n_{Y,1} : \prod(x : \mathrm{Fin}(A)) \prod(a : Y[x]), \cup_Y\, x\, \emptyset\, (a, \emptyset_Y) =^Y_{\mathrm{neut}_1} a \\ n_{Y,2} : \prod(x : \mathrm{Fin}(A)) \prod(a : Y[x]), \cup_Y\, \emptyset\, x\, (\emptyset_Y, a) =^Y_{\mathrm{neut}_2} a \\ c_Y : \prod(x, y : \mathrm{Fin}(A)) \prod(a : Y[x]) \prod(b : Y[y]), \\ \cup_Y\, x\, y (a, b) =^Y_{\mathrm{com}} \cup_Y\, y\, x\, (b, a) \\ i_Y : \prod(a : A), \cup_Y\, (L\, a)\, (L\, a)\, (L_Y\, x, L_Y\, x) =^Y_{\mathrm{idem}} L_Y\, x \end{array}}{\mathrm{Fin}(A)\text{-rec}(\emptyset_Y, L_y, \cup_Y, a_Y, n_{Y,1}, n_{Y,2}, c_Y, i_Y) : \prod(x : \mathrm{Fin}(A)), Y}$$

Elimination Rule for Kuratowski Sets

To make it more readable, we remove the fibers.

$$\frac{\begin{array}{c} Y : \mathrm{Fin}(A) \to \mathit{Type} \\ \emptyset_Y : Y[\emptyset] \\ L_Y : \prod(a : A), Y[L\ a] \\ \cup_Y : \prod(x, y : \mathrm{Fin}\ A), Y[x] \times Y[y] \to Y[\cup(x, y)] \\ a_Y : \prod(x, y, z : \mathrm{Fin}(A)) \prod(a : Y[x]) \prod(b : Y[y]) \prod(c : Y[z]), \\ \cup_Y(a, (\cup_Y(b, c))) =^Y_{\mathrm{as}} \cup_Y(\cup_Y(a, b), c) \\ n_{Y,1} : \prod(x : \mathrm{Fin}(A)) \prod(a : Y[x]), \cup_Y(a, \emptyset_Y) =^Y_{\mathrm{neut}_1} a \\ n_{Y,2} : \prod(x : \mathrm{Fin}(A)) \prod(a : Y[x]), \cup_Y(\emptyset_Y, a) =^Y_{\mathrm{neut}_2} a \\ c_Y : \prod(x, y : \mathrm{Fin}(A)) \prod(a : Y[x]) \prod(b : Y[y]), \\ \cup_Y(a, b) =^Y_{\mathrm{com}} \cup_Y(b, a) \\ i_Y : \prod(a : A), \cup_Y(L_Y\ x, L_Y\ x) =^Y_{\mathrm{idem}} L_Y\ x \end{array}}{\mathrm{Fin}(A)\text{-rec}(\emptyset_Y, L_y, \cup_Y, a_Y, n_{Y,1}, n_{Y,2}, c_Y, i_Y) : \prod(x : \mathrm{Fin}(A)), Y}$$

Conclusion and Further Work

- Higher inductive types offer good opportunities for programming. Closer to specification.
- Some further work: add higher paths, good formal semantics.