The Gentle Art of Levitation

James Chapman[†] Pierre-Évariste Dagand[‡]
Conor McBride[‡] Peter Morris[*]

Institute of Cybernetics, Tallinn University of Technology[†]

University of Strathclyde[‡]

University of Nottingham[*]

September 27, 2010

Inductive datatypes

An inductive type

```haskell
data List z = Nil | Cons z (List z)
```

The essence of inductive types

```haskell
data ListF z list = Nil | Cons z list
    deriving Functor
```

$$ListF\ z\ list = 1 + z \times list$$

```haskell
newtype Mu f = In { in :: f (Mu f) }
```

```haskell
type List z = Mu (ListF z)
```

The Power of Sigma!

Datatypes in ML

- "sum-of-products"
- $\mathit{ListF}\ z\ \mathit{list} = 1 + z \times \mathit{list}$

Dependent types?

- Σ for sum: $\Sigma_{x:S}\, T$
- Σ for product: $(x : S) \times T$

Desc: a universe of datatypes

$$\begin{array}{ll}
\mathsf{Desc} & : \textsc{Set} \\
 & \\
\text{'}1 & : \mathsf{Desc} \\
\text{'}\Sigma\ (S : \textsc{Set})\ (D : S \to \mathsf{Desc}) & : \mathsf{Desc} \\
\text{'}\mathsf{ind}{\times}\ (D : \mathsf{Desc}) & : \mathsf{Desc}
\end{array}
\qquad
\begin{array}{ll}
[\![_]\!] & : \mathsf{Desc} \to \textsc{Set} \to \textsc{Set} \\
 & \\
[\![\text{'}1]\!]\ X & \mapsto 1 \\
[\![\text{'}\Sigma\ S\ D]\!]\ X & \mapsto (s : S) \times [\![D\ s]\!] X \\
[\![\text{'}\mathsf{ind}{\times}\ D]\!]\ X & \mapsto X \times [\![D]\!]\ X
\end{array}$$

Example

The universe of datatypes

$$
\begin{array}{lll}
\text{'}1 & : \mathrm{Desc} & [\![\text{'}1]\!]\ X \mapsto 1 \\
\text{'}\Sigma\ (S : \textsc{Set})\ (D : S \to \mathrm{Desc}) & : \mathrm{Desc} & [\![\text{'}\Sigma\ S\ D]\!]\ X \mapsto (s : S) \times [\![D\ s]\!] X \\
\text{'ind}\times\ (D : \mathrm{Desc}) & : \mathrm{Desc} & [\![\text{'ind}\times\ D]\!]\ X \mapsto X \times [\![D]\!]\ X
\end{array}
$$

$$ListF\ z\ list = 1 + z \times list$$

Pattern functor of List

$$
\begin{array}{l}
\mathrm{ListD} : \textsc{Set} \to \mathrm{Desc} \\
\mathrm{ListD}\ Z \mapsto \boxed{\{\mathrm{Desc}\}}
\end{array}
$$

Example

The universe of datatypes

$$
\begin{array}{lll}
\text{'}1 & : \mathsf{Desc} & [\![\text{'}1]\!]\ X \mapsto 1 \\
\text{'}\Sigma\ (S : \textsc{Set})\ (D : S \to \mathsf{Desc}) & : \mathsf{Desc} & [\![\text{'}\Sigma\ S\ D]\!]\ X \mapsto (s : S) \times [\![D\ s]\!] X \\
\text{'ind}\times\ (D : \mathsf{Desc}) & : \mathsf{Desc} & [\![\text{'ind}\times\ D]\!]\ X \mapsto X \times [\![D]\!]\ X
\end{array}
$$

$$\mathit{ListF}\ z\ \mathit{list} = 1 + z \times \mathit{list}$$

Pattern functor of List

$$
\begin{array}{l}
\mathsf{ListD} : \textsc{Set} \to \mathsf{Desc} \\
\mathsf{ListD}\ Z \mapsto \text{'}\Sigma\ \{?S : \textsc{Set}\}\ \{?S \to \mathsf{Desc}\}
\end{array}
$$

Example

The universe of datatypes

$$
\begin{array}{llll}
\text{'1} & : \mathsf{Desc} & [\![\text{'1}]\!]\ X & \mapsto 1 \\
\text{'}\Sigma\ (S : \mathrm{SET})\ (D : S \to \mathsf{Desc}) & : \mathsf{Desc} & [\![\text{'}\Sigma\ S\ D]\!]\ X & \mapsto (s : S) \times [\![D\ s]\!] X \\
\text{'ind}\times\ (D : \mathsf{Desc}) & : \mathsf{Desc} & [\![\text{'ind}\times\ D]\!]\ X & \mapsto X \times [\![D]\!]\ X
\end{array}
$$

$$
\mathit{ListF}\ z\ \mathit{list} = 1 + z \times \mathit{list}
$$

Pattern functor of List

$$
\mathsf{ListD} : \mathrm{SET} \to \mathsf{Desc}
$$

$$
\mathsf{ListD}\ Z \mapsto \text{'}\Sigma\ \#\begin{bmatrix} \text{'nil} \\ \text{'cons} \end{bmatrix}\ \boxed{\{\#\begin{bmatrix} \vdots \end{bmatrix} \to \mathsf{Desc}\}}
$$

Example

The universe of datatypes

$$
\begin{array}{ll}
\text{'1} : \mathsf{Desc} & [\![\text{'1}]\!]\ X \mapsto 1 \\
\text{'}\Sigma\ (S : \mathrm{SET})\ (D : S \to \mathsf{Desc}) : \mathsf{Desc} & [\![\text{'}\Sigma\ S\ D]\!]\ X \mapsto (s : S) \times [\![D\ s]\!] X \\
\text{'ind}\times\ (D : \mathsf{Desc}) : \mathsf{Desc} & [\![\text{'ind}\times\ D]\!]\ X \mapsto X \times [\![D]\!]\ X
\end{array}
$$

$$ListF\ z\ list = 1 + z \times list$$

Pattern functor of List

$$\mathsf{ListD} : \mathrm{SET} \to \mathsf{Desc}$$

$$\mathsf{ListD}\ Z \mapsto \text{'}\Sigma\ \#\begin{bmatrix} \text{'nil} \\ \text{'cons} \end{bmatrix} \begin{bmatrix} \boxed{\{\mathsf{Desc}\}} \\ \boxed{\{\mathsf{Desc}\}} \end{bmatrix}^{\lambda\_.\ \mathsf{Desc}}$$

Example

The universe of datatypes

$$
\begin{array}{lll}
\text{'1} & : \mathrm{Desc} & [\![\text{'1}]\!]\ X \mapsto 1 \\
\text{'}\Sigma\ (S : \mathrm{SET})\ (D : S \to \mathrm{Desc}) & : \mathrm{Desc} & [\![\text{'}\Sigma\ S\ D]\!]\ X \mapsto (s : S) \times [\![D\ s]\!] X \\
\text{'ind}\times\ (D : \mathrm{Desc}) & : \mathrm{Desc} & [\![\text{'ind}\times\ D]\!]\ X \mapsto X \times [\![D]\!]\ X
\end{array}
$$

$$ListF\ z\ list = 1 + z \times list$$

Pattern functor of List

$$\mathrm{ListD} : \mathrm{SET} \to \mathrm{Desc}$$

$$\mathrm{ListD}\ Z \mapsto \text{'}\Sigma\ \#\begin{bmatrix} \text{'nil} \\ \text{'cons} \end{bmatrix} \begin{bmatrix} \text{'1} \\ \{\mathrm{Desc}\} \end{bmatrix}^{\lambda\_.\ \mathrm{Desc}}$$

Example

The universe of datatypes

$$
\begin{array}{lll}
\text{'1} & : \mathsf{Desc} & [\![\text{'1}]\!]\, X \mapsto 1 \\
\text{'}\Sigma\, (S : \mathrm{SET})\, (D : S \to \mathsf{Desc}) & : \mathsf{Desc} & [\![\text{'}\Sigma\; S\; D]\!]\, X \mapsto (s : S) \times [\![D\; s]\!] X \\
\text{'ind}\times\, (D : \mathsf{Desc}) & : \mathsf{Desc} & [\![\text{'ind}\times\; D]\!]\, X \mapsto X \times [\![D]\!]\, X
\end{array}
$$

$$\mathit{ListF}\; z\; \mathit{list} = 1 + z \times \mathit{list}$$

Pattern functor of List

$$\mathsf{ListD} : \mathrm{SET} \to \mathsf{Desc}$$

$$\mathsf{ListD}\; Z \mapsto \text{'}\Sigma\; \# \begin{bmatrix} \text{'nil} \\ \text{'cons} \end{bmatrix} \begin{bmatrix} \text{'1} \\ \text{'}\Sigma\; Z\; (\lambda\_ .\; \boxed{\{\mathsf{Desc}\}}) \end{bmatrix}^{\lambda\_ .\; \mathsf{Desc}}$$

Example

The universe of datatypes

$$
\begin{array}{lll}
\text{'}1 & : \mathsf{Desc} & [\![\text{'}1]\!]\, X \mapsto 1 \\
\text{'}\Sigma\,(S : \mathrm{SET})\,(D : S \to \mathsf{Desc}) & : \mathsf{Desc} & [\![\text{'}\Sigma\; S\; D]\!]\, X \mapsto (s : S) \times [\![D\; s]\!] X \\
\text{'ind}\times\,(D : \mathsf{Desc}) & : \mathsf{Desc} & [\![\text{'ind}\times\; D]\!]\, X \mapsto X \times [\![D]\!]\, X
\end{array}
$$

$$
\mathit{ListF}\; z\; \mathit{list} = 1 + z \times \mathit{list}
$$

Pattern functor of List

$$
\begin{array}{l}
\mathsf{ListD} : \mathrm{SET} \to \mathsf{Desc} \\
\mathsf{ListD}\; Z \mapsto \text{'}\Sigma\; \#\begin{bmatrix} \text{'nil} \\ \text{'cons} \end{bmatrix} \begin{bmatrix} \text{'}1 \\ \text{'}\Sigma\; Z\; (\lambda\_.\; \text{'ind}\times\; \text{'}1) \end{bmatrix}^{\lambda\_.\; \mathsf{Desc}}
\end{array}
$$

Fixpoint and induction

Fix-point of Descriptions

$$\frac{\Gamma \vdash D : \mathsf{Desc}}{\Gamma \vdash \mu D : \textsc{Set}} \qquad \frac{\Gamma \vdash D : \mathsf{Desc} \quad \Gamma \vdash d : [\![D]\!]\ (\mu D)}{\Gamma \vdash \mathsf{con}\ d : \mu D}$$

Induction principle

$$\begin{aligned}\mathsf{ind} :\ & (D : \mathsf{Desc})(P : \mu D \to \textsc{Set}) \to \\ & ((d : [\![D]\!]\ (\mu D)) \to \mathsf{All}\ D\ (\mu D)\ P\ d \to P(\mathsf{con}\ d)) \to \\ & (x : \mu D) \to P x\end{aligned}$$

$$\mathsf{List}\ Z \mapsto \mu(\mathsf{ListD}\ Z)$$

Another example?

The code of our universe

$$\begin{array}{ll} \mathsf{Desc} & : \textsc{Set} \\ \text{'}1 & : \mathsf{Desc} \\ \text{'}\Sigma\ (S : \textsc{Set})\ (D : S \to \mathsf{Desc}) & : \mathsf{Desc} \\ \text{'}\mathsf{ind}\!\times\ (D : \mathsf{Desc}) & : \mathsf{Desc} \end{array}$$

It is nothing but an inductive type!

Let's *describe* it!

$$\begin{array}{l} \mathsf{DescD} : \mathsf{Desc} \\ \mathsf{DescD} \mapsto \boxed{\{\mathsf{Desc}\}} \end{array}$$

Another example?

The code of our universe

$$\begin{array}{ll}
\mathsf{Desc} & : \textsc{Set} \\
\text{'}1 & : \mathsf{Desc} \\
\text{'}\Sigma\ (S : \textsc{Set})\ (D : S \to \mathsf{Desc}) & : \mathsf{Desc} \\
\text{'}\mathsf{ind}\!\times\ (D : \mathsf{Desc}) & : \mathsf{Desc}
\end{array}$$

It is nothing but an inductive type!

Let's *describe* it!

$$\begin{array}{l}
\mathsf{DescD} : \mathsf{Desc} \\
\mathsf{DescD} \mapsto \text{'}\Sigma\ \boxed{\{?S : \textsc{Set}\}}\ \boxed{\{?S \to \mathsf{Desc}\}}
\end{array}$$

Another example?

The code of our universe

$$
\begin{array}{ll}
\mathsf{Desc} & : \textsc{Set} \\
\text{'}1 & : \mathsf{Desc} \\
\text{'}\Sigma\ (S : \textsc{Set})\ (D : S \to \mathsf{Desc}) & : \mathsf{Desc} \\
\text{'ind}\!\times\ (D : \mathsf{Desc}) & : \mathsf{Desc}
\end{array}
$$

It is nothing but an inductive type!

Let's *describe* it!

$$\mathsf{DescD} : \mathsf{Desc}$$

$$\mathsf{DescD} \mapsto \text{'}\Sigma\ \#\begin{bmatrix} \text{'}1 \\ \text{'}\Sigma \\ \text{'ind}\!\times \end{bmatrix} \begin{bmatrix} \{\mathsf{Desc}\} \\ \{\mathsf{Desc}\} \\ \{\mathsf{Desc}\} \end{bmatrix}^{\lambda\_.\ \mathsf{Desc}}$$

Another example?

The code of our universe

$$\begin{array}{ll}
\mathsf{Desc} & : \textsc{Set} \\
\text{'1} & : \mathsf{Desc} \\
\text{'}\Sigma\ (S : \textsc{Set})\ (D : S \to \mathsf{Desc}) & : \mathsf{Desc} \\
\text{'ind}\times\ (D : \mathsf{Desc}) & : \mathsf{Desc}
\end{array}$$

It is nothing but an inductive type!

Let's *describe* it!

$$\mathsf{DescD} : \mathsf{Desc}$$

$$\mathsf{DescD} \mapsto \text{'}\Sigma\ \#\begin{bmatrix} \text{'1} \\ \text{'}\Sigma \\ \text{'ind}\times \end{bmatrix} \begin{bmatrix} \text{'1} \\ \boxed{\{\mathsf{Desc}\}} \\ \boxed{\{\mathsf{Desc}\}} \end{bmatrix}^{\lambda\_.\ \mathsf{Desc}}$$

Another example?

The code of our universe

$$\begin{array}{ll}
\mathsf{Desc} & : \mathrm{S{\scriptsize ET}} \\
\text{'}1 & : \mathsf{Desc} \\
\text{'}\Sigma\ (S : \mathrm{S{\scriptsize ET}})\ (D : S \to \mathsf{Desc}) & : \mathsf{Desc} \\
\text{'ind}\times\ (D : \mathsf{Desc}) & : \mathsf{Desc}
\end{array}$$

It is nothing but an inductive type!

Let's *describe* it!

$$\mathsf{DescD} : \mathsf{Desc}$$

$$\mathsf{DescD} \mapsto \text{'}\Sigma\ \#\begin{bmatrix} \text{'}1 \\ \text{'}\Sigma \\ \text{'ind}\times \end{bmatrix} \begin{bmatrix} \text{'}1 \\ \boxed{\{\mathsf{Desc}\}} \\ \text{'ind}\times\ \text{'}1 \end{bmatrix}^{\lambda\_.\ \mathsf{Desc}}$$

Another example?

The code of our universe

$$\begin{array}{ll}
\mathsf{Desc} & : \textsc{Set} \\
\text{'1} & : \mathsf{Desc} \\
\text{'}\Sigma\ (S : \textsc{Set})\ (D : S \to \mathsf{Desc}) & : \mathsf{Desc} \\
\text{'ind}\times\ (D : \mathsf{Desc}) & : \mathsf{Desc}
\end{array}$$

It is nothing but an inductive type!

Let's *describe* it!

$$\mathsf{DescD} : \mathsf{Desc}$$

$$\mathsf{DescD} \mapsto \text{'}\Sigma\ \# \begin{bmatrix} \text{'1} \\ \text{'}\Sigma \\ \text{'ind}\times \end{bmatrix} \begin{bmatrix} \text{'1} \\ \text{'}\Sigma\ \textsc{Set}\ \lambda S.\ \boxed{\{\mathsf{Desc}\}} \\ \text{'ind}\times\ \text{'1} \end{bmatrix}^{\lambda\_.\ \mathsf{Desc}}$$

Second attempt

Extending the universe

$$
\begin{array}{ll}
\text{'1} & : \mathsf{Desc} \\
\text{'}\Sigma\ (S : \mathrm{SET})\ (D : S \to \mathsf{Desc}) & : \mathsf{Desc} \\
\text{'ind}\times\ (D : \mathsf{Desc}) & : \mathsf{Desc} \\
\text{'hind}\times\ (H : \mathrm{SET})\ (D : \mathsf{Desc}) & : \mathsf{Desc} \quad [\![\text{'hind}\times\ H\ D]\!]\ X \quad \mapsto \quad (H \to X) \times [\![D]\!]\ X
\end{array}
$$

Let's try again!

$$\mathsf{DescD} : \mathsf{Desc}$$

$$\mathsf{DescD} \mapsto \text{'}\Sigma\ \# \begin{bmatrix} \text{'1} \\ \text{'}\Sigma \\ \text{'ind}\times \\ \text{'hind}\times \end{bmatrix} \begin{bmatrix} \text{'1} \\ \text{'}\Sigma\ \mathrm{SET}\ \lambda S.\ \boxed{\{\mathsf{Desc}\}} \\ \text{'ind}\times\ \text{'1} \\ \boxed{\{\mathsf{Desc}\}} \end{bmatrix}^{\lambda\_.\ \mathsf{Desc}}$$

Second attempt

Extending the universe

$$
\begin{array}{ll}
\text{'1} & : \mathsf{Desc} \\
\text{'}\Sigma\ (S : \textsc{Set})\ (D : S \to \mathsf{Desc}) & : \mathsf{Desc} \\
\text{'ind}\times\ (D : \mathsf{Desc}) & : \mathsf{Desc} \\
\text{'hind}\times\ (H : \textsc{Set})\ (D : \mathsf{Desc}) & : \mathsf{Desc} \quad [\![\text{'hind}\times\ H\ D]\!]\ X \quad \mapsto\ (H \to X) \times [\![D]\!]\ X
\end{array}
$$

Let's try again!

$$
\begin{array}{l}
\mathsf{DescD} : \mathsf{Desc} \\
\mathsf{DescD} \mapsto \text{'}\Sigma\ \#\begin{bmatrix} \text{'1} \\ \text{'}\Sigma \\ \text{'ind}\times \\ \text{'hind}\times \end{bmatrix} \begin{bmatrix} \text{'1} \\ \text{'}\Sigma\ \textsc{Set}\ \lambda S.\ \text{'hind}\times\ S\ \text{'1} \\ \text{'ind}\times\ \text{'1} \\ \boxed{\{\mathsf{Desc}\}} \end{bmatrix}^{\lambda\_.\ \mathsf{Desc}}
\end{array}
$$

Second attempt

Extending the universe

$$\begin{array}{ll}
\text{'1} & : \mathsf{Desc} \\
\text{'}\Sigma\ (S : \textsc{Set})\ (D : S \to \mathsf{Desc}) & : \mathsf{Desc} \\
\text{'ind}\times\ (D : \mathsf{Desc}) & : \mathsf{Desc} \\
\text{'hind}\times\ (H : \textsc{Set})\ (D : \mathsf{Desc}) & : \mathsf{Desc} \quad [\![\text{'hind}\times\ H\ D]\!]\ X \quad \mapsto \quad (H \to X) \times [\![D]\!]\ X
\end{array}$$

Let's try again!

$$\mathsf{DescD} : \mathsf{Desc}$$

$$\mathsf{DescD} \mapsto \text{'}\Sigma\ \#\begin{bmatrix} \text{'1} \\ \text{'}\Sigma \\ \text{'ind}\times \\ \text{'hind}\times \end{bmatrix} \begin{bmatrix} \text{'1} \\ \text{'}\Sigma\ \textsc{Set}\ \lambda S.\ \text{'hind}\times\ S\ \text{'1} \\ \text{'ind}\times\ \text{'1} \\ \text{'}\Sigma\ \textsc{Set}\ \lambda\_.\ \boxed{\{\mathsf{Desc}\}} \end{bmatrix}^{\lambda\_.\ \mathsf{Desc}}$$

Second attempt

Extending the universe

$$
\begin{array}{ll}
\text{'1} & : \mathsf{Desc} \\
\text{'}\Sigma\ (S : \textsc{Set})\ (D : S \to \mathsf{Desc}) & : \mathsf{Desc} \\
\text{'ind}\times\ (D : \mathsf{Desc}) & : \mathsf{Desc} \\
\text{'hind}\times\ (H : \textsc{Set})\ (D : \mathsf{Desc}) & : \mathsf{Desc} \quad [\![\text{'hind}\times\ H\ D]\!]\ X \quad \mapsto \quad (H \to X) \times [\![D]\!]\ X
\end{array}
$$

Let's try again!

$$\mathsf{DescD} : \mathsf{Desc}$$

$$
\mathsf{DescD} \mapsto \text{'}\Sigma\ \#\begin{bmatrix} \text{'1} \\ \text{'}\Sigma \\ \text{'ind}\times \\ \text{'hind}\times \end{bmatrix} \begin{bmatrix} \text{'1} \\ \text{'}\Sigma\ \textsc{Set}\ \lambda S.\ \text{'hind}\times\ S\ \text{'1} \\ \text{'ind}\times\ \text{'1} \\ \text{'}\Sigma\ \textsc{Set}\ \lambda\_.\ \text{'ind}\times\ \text{'1} \end{bmatrix}^{\lambda\_.\ \mathsf{Desc}}
$$

Levitation!

The problem

$$\mathsf{DescD} \mapsto {}'\Sigma\ \#\begin{bmatrix} '1 \\ '\Sigma \\ '\mathsf{ind}\times \\ '\mathsf{hind}\times \end{bmatrix}\begin{bmatrix} '1 \\ '\Sigma\ \mathrm{SET}\ \lambda S.\ '\mathsf{hind}\times\ S\ '1 \\ '\mathsf{ind}\times\ '1 \\ '\Sigma\ \mathrm{SET}\ \lambda\_.\ '\mathsf{ind}\times\ '1 \end{bmatrix}^{\lambda\_.\ \mathsf{Desc}}$$
$$\mathsf{Desc} \mapsto \mu\mathsf{DescD}$$

One solution

$$\mathsf{DescD} \mapsto {}'\Sigma\ \#\begin{bmatrix} '1 \\ '\Sigma \\ '\mathsf{ind}\times \\ '\mathsf{hind}\times \end{bmatrix}\begin{Bmatrix} '1 \\ '\Sigma\ \mathrm{SET}\ \lambda S.\ '\mathsf{hind}\times\ S\ '1 \\ '\mathsf{ind}\times\ '1 \\ '\Sigma\ \mathrm{SET}\ \lambda\_.\ '\mathsf{ind}\times\ '1 \end{Bmatrix}$$
$$\mathsf{Desc} \mapsto \mu\mathsf{DescD}$$

Levitation!

The problem

$$\mathsf{DescD} \mapsto \text{'}\Sigma\ \# \left[\begin{array}{l} \text{'1} \\ \text{'}\Sigma \\ \text{'ind}\times \\ \text{'hind}\times \end{array}\right] \left[\begin{array}{l} \text{'1} \\ \text{'}\Sigma\ \textsc{Set}\ \lambda S.\ \text{'hind}\times\ S\ \text{'1} \\ \text{'ind}\times\ \text{'1} \\ \text{'}\Sigma\ \textsc{Set}\ \lambda\_.\ \text{'ind}\times\ \text{'1} \end{array}\right]^{\lambda\_.\ \boxed{\mathsf{Desc}}}$$

$$\mathsf{Desc} \mapsto \mu\mathsf{DescD}$$

One solution

$$\mathsf{DescD} \mapsto \text{'}\Sigma\ \# \left[\begin{array}{l} \text{'1} \\ \text{'}\Sigma \\ \text{'ind}\times \\ \text{'hind}\times \end{array}\right] \left\{\begin{array}{l} \text{'1} \\ \text{'}\Sigma\ \textsc{Set}\ \lambda S.\ \text{'hind}\times\ S\ \text{'1} \\ \text{'ind}\times\ \text{'1} \\ \text{'}\Sigma\ \textsc{Set}\ \lambda\_.\ \text{'ind}\times\ \text{'1} \end{array}\right\}$$

$$\mathsf{Desc} \mapsto \mu\mathsf{DescD}$$

Stepping back

The implementation

Object	Role	Status
Desc	Describe pattern functors	Levitated
$[\![\_]\!]$	Interpret descriptions	Hardwired
μ, con	Define, inhabit fixpoints	Hardwired
ind, All, all	Induction principle	Hardwired

Consequences

- Closed, non-generative presentation of datatypes
- Desc is plain data
- Desc comes with an induction principle, for free

Generic programing is *just* programming!

Generic programming in action

The catamorphism

$$\mathsf{cata} : (D : \mathsf{Desc})(T : \mathrm{S}\scriptstyle{\mathrm{ET}}\displaystyle) \to (\llbracket D \rrbracket\ T \to T) \to \mu D \to T$$

$$\mathsf{cata}\ D\ T\ f \mapsto \mathsf{ind}\ \{?D : \mathsf{Desc}\}\ \{?P : \mu ?D \to \mathrm{S}\scriptstyle{\mathrm{ET}}\displaystyle\}$$
$$\{(d : \llbracket ?D \rrbracket\ (\mu ?D)) \to \mathsf{All}\ ?D\ (\mu ?D)\ ?P\ d \to ?P(\mathsf{con}\ d)\}$$

Generic programming in action

The catamorphism

$$\mathsf{cata} : (D : \mathsf{Desc})(T : \mathrm{SET}) \to (\llbracket D \rrbracket\ T \to T) \to \mu D \to T$$
$$\begin{aligned}\mathsf{cata}\ D\ T\ f \mapsto \mathsf{ind}\ D\ & \{?P : \mu D \to \mathrm{SET}\} \\ & \{(d : \llbracket D \rrbracket\ (\mu D)) \to \mathsf{All}\ D\ (\mu D)\ ?P\ d \to ?P(\mathsf{con}\ d)\}\end{aligned}$$

Generic programming in action

The catamorphism

$$
\begin{array}{l}
\mathsf{cata} : (D : \mathsf{Desc})(T : \mathrm{SET}) \to (\llbracket D \rrbracket\ T \to T) \to \mu D \to T \\
\mathsf{cata}\ D\ T\ f \mapsto \mathsf{ind}\ D\ (\lambda\_.\ T) \\
\qquad\qquad\qquad\qquad \{(d : \llbracket D \rrbracket\ (\mu D)) \to \mathsf{All}\ D\ (\mu D)\ (\lambda\_.\ T)\ d \to T\}
\end{array}
$$

Generic programming in action

The catamorphism

$$\mathsf{cata} : (D : \mathsf{Desc})(T : \mathrm{SET}) \to (\llbracket D \rrbracket\ T \to T) \to \mu D \to T$$
$$\mathsf{cata}\ D\ T\ f \mapsto \mathsf{ind}\ D\ (\lambda\_.\ T)$$
$$(\lambda xs.\ \lambda hs.\ \{T\})$$

Generic programming in action

The catamorphism

$$
\begin{aligned}
&\mathsf{cata} : (D : \mathsf{Desc})(T : \mathrm{SET}) \to (\llbracket D \rrbracket\ T \to T) \to \mu D \to T \\
&\mathsf{cata}\ D\ T\ f \mapsto \mathsf{ind}\ D\ (\lambda\_.\ T) \\
&\qquad\qquad\qquad\qquad (\lambda xs.\ \lambda hs.\ f\ \{\llbracket D \rrbracket\ T\})
\end{aligned}
$$

Generic programming in action

The catamorphism

$$\mathsf{cata} : (D\!:\!\mathsf{Desc})(T\!:\!\mathrm{SET}) \to (\llbracket D \rrbracket\ T \to T) \to \mu D \to T$$
$$\mathsf{cata}\ D\ T\ f \mapsto \mathsf{ind}\ D\ (\lambda\_.\ T)\ (\lambda xs.\ \lambda hs.\ f\ (\mathsf{replace}\ D\ \mu D\ T\ xs\ hs))$$

$$\begin{array}{llll}
\mathsf{replace} : (D\!:\!\mathsf{Desc})(X, Y\!:\!\mathrm{SET})(xs\!:\!\llbracket D \rrbracket\ X) \to \mathsf{All}\ D\ X\ (\lambda_.\ Y)\ xs \to \llbracket D \rrbracket\ Y & & & \\
\mathsf{replace}\ \text{'}1 & X\ Y\ []\quad [] & \mapsto & \{1\} \\
\mathsf{replace}\ (\text{'}\Sigma\ S\ D) & X\ Y\ [s, d]\ d' & \mapsto & \{(s\!:\!S) \times \llbracket D\ s \rrbracket\ Y\} \\
\mathsf{replace}\ (\text{'ind}\times\ D) & X\ Y\ [x, d]\ [y, d'] & \mapsto & \{Y \times \llbracket D \rrbracket\ Y\} \\
\mathsf{replace}\ (\text{'hind}\times\ H\ D) & X\ Y\ [f, d]\ [g, d'] & \mapsto & \{H \to Y \times \llbracket D \rrbracket\ Y\}
\end{array}$$

Generic programming in action

The catamorphism

$$
\begin{aligned}
&\mathsf{cata} : (D : \mathsf{Desc})(T : \mathrm{SET}) \to (\llbracket D \rrbracket\ T \to T) \to \mu D \to T \\
&\mathsf{cata}\ D\ T\ f \mapsto \mathsf{ind}\ D\ (\lambda \_ .\ T) \\
&\qquad\qquad\qquad\qquad (\lambda xs.\ \lambda hs.\ f\ (\mathsf{replace}\ D\ \mu D\ T\ xs\ hs))
\end{aligned}
$$

$$
\begin{array}{llllll}
\multicolumn{6}{l}{\mathsf{replace} : (D : \mathsf{Desc})(X, Y : \mathrm{SET})(xs : \llbracket D \rrbracket\ X) \to \mathsf{All}\ D\ X\ (\lambda \_ .\ Y)\ xs \to \llbracket D \rrbracket\ Y} \\
\mathsf{replace}\ \text{'1} & X\ Y & [] & [] & \mapsto & [] \\
\mathsf{replace}\ (\text{'}\Sigma\ S\ D) & X\ Y & [s, d] & d' & \mapsto & \boxed{\{(s : S) \times \llbracket D\ s \rrbracket\ Y\}} \\
\mathsf{replace}\ (\text{'ind}\times\ D) & X\ Y & [x, d] & [y, d'] & \mapsto & \boxed{\{Y \times \llbracket D \rrbracket\ Y\}} \\
\mathsf{replace}\ (\text{'hind}\times\ H\ D) & X\ Y & [f, d] & [g, d'] & \mapsto & \boxed{\{H \to Y \times \llbracket D \rrbracket\ Y\}}
\end{array}
$$

Generic programming in action

The catamorphism

$$
\begin{array}{l}
\mathsf{cata} : (D : \mathsf{Desc})(T : \mathrm{SET}) \to (\llbracket D \rrbracket\ T \to T) \to \mu D \to T \\
\mathsf{cata}\ D\ T\ f \mapsto \mathsf{ind}\ D\ (\lambda\_.\ T) \\
\qquad\qquad\qquad\qquad (\lambda xs.\ \lambda hs.\ f\ (\mathsf{replace}\ D\ \mu D\ T\ xs\ hs))
\end{array}
$$

$$
\begin{array}{lllll}
\multicolumn{5}{l}{\mathsf{replace} : (D : \mathsf{Desc})(X, Y : \mathrm{SET})(xs : \llbracket D \rrbracket\ X) \to \mathsf{All}\ D\ X\ (\lambda\_.\ Y)\ xs \to \llbracket D \rrbracket\ Y} \\
\mathsf{replace}\ \text{'}1 & X\ Y\ [] & [] & \mapsto & [] \\
\mathsf{replace}\ (\text{'}\Sigma\ S\ D) & X\ Y\ [s, d] & d' & \mapsto & [s, \mathsf{replace}\ (D\ s)\ X\ Y\ d\ d'] \\
\mathsf{replace}\ (\text{'ind}\times\ D) & X\ Y\ [x, d] & [y, d'] & \mapsto & \{Y \times \llbracket D \rrbracket\ Y\} \\
\mathsf{replace}\ (\text{'hind}\times\ H\ D) & X\ Y\ [f, d] & [g, d'] & \mapsto & \{H \to Y \times \llbracket D \rrbracket\ Y\}
\end{array}
$$

Generic programming in action

The catamorphism

$$\mathsf{cata} : (D : \mathsf{Desc})(T : \mathrm{SET}) \to (\llbracket D \rrbracket\ T \to T) \to \mu D \to T$$
$$\mathsf{cata}\ D\ T\ f \mapsto \mathsf{ind}\ D\ (\lambda\_.\ T)$$
$$\qquad (\lambda xs.\ \lambda hs.\ f\ (\mathsf{replace}\ D\ \mu D\ T\ xs\ hs))$$

$$\mathsf{replace} : (D : \mathsf{Desc})(X, Y : \mathrm{SET})(xs : \llbracket D \rrbracket\ X) \to \mathsf{All}\ D\ X\ (\lambda\_.\ Y)\ xs \to \llbracket D \rrbracket\ Y$$
$$\mathsf{replace}\ \text{'}1\ X\ Y\ []\ []\ \mapsto\ []$$
$$\mathsf{replace}\ (\text{'}\Sigma\ S\ D)\ X\ Y\ [s, d]\ d'\ \mapsto\ [s, \mathsf{replace}\ (D\ s)\ X\ Y\ d\ d']$$
$$\mathsf{replace}\ (\text{'ind}\times\ D)\ X\ Y\ [x, d]\ [y, d']\ \mapsto\ [y, \mathsf{replace}\ D\ X\ Y\ d\ d']$$
$$\mathsf{replace}\ (\text{'hind}\times\ H\ D)\ X\ Y\ [f, d]\ [g, d']\ \mapsto\ \{H \to Y \times \llbracket D \rrbracket\ Y\}$$

Generic programming in action

The catamorphism

$$
\begin{array}{l}
\mathsf{cata} : (D : \mathsf{Desc})(T : \mathrm{SET}) \to (\llbracket D \rrbracket\ T \to T) \to \mu D \to T \\
\mathsf{cata}\ D\ T\ f \mapsto \mathsf{ind}\ D\ (\lambda\_.\ T) \\
\qquad\qquad\qquad\qquad (\lambda xs.\ \lambda hs.\ f\ (\mathsf{replace}\ D\ \mu D\ T\ xs\ hs))
\end{array}
$$

$$
\begin{array}{llllll}
\multicolumn{6}{l}{\mathsf{replace} : (D : \mathsf{Desc})(X, Y : \mathrm{SET})(xs : \llbracket D \rrbracket\ X) \to \mathsf{All}\ D\ X\ (\lambda\_.\ Y)\ xs \to \llbracket D \rrbracket\ Y} \\
\mathsf{replace}\ \text{'}1 & X\ Y\ [] & [] & \mapsto & [] \\
\mathsf{replace}\ (\text{'}\Sigma\ S\ D) & X\ Y\ [s, d] & d' & \mapsto & [s, \mathsf{replace}\ (D\ s)\ X\ Y\ d\ d'] \\
\mathsf{replace}\ (\text{'ind}\times\ D) & X\ Y\ [x, d] & [y, d'] & \mapsto & [y, \mathsf{replace}\ D\ X\ Y\ d\ d'] \\
\mathsf{replace}\ (\text{'hind}\times\ H\ D) & X\ Y\ [f, d] & [g, d'] & \mapsto & [g, \mathsf{replace}\ D\ X\ Y\ d\ d']
\end{array}
$$

Generic programming in action

The Free Monad

Tagged descriptions

$$\mathsf{TagDesc} : \textsc{Set}$$
$$\mathsf{TagDesc} \mapsto ([\ldots] : \mathsf{En}) \times ([\ldots]^{\mathsf{Desc}})$$

$$\mathsf{de} : \mathsf{TagDesc} \to \mathsf{Desc} \qquad \mathsf{NatD} \mapsto \mathsf{de} \left[\begin{bmatrix} \text{'zero} \\ \text{'suc} \end{bmatrix}, \begin{bmatrix} \text{'1} \\ \text{'ind}\times \text{'1} \end{bmatrix} \right]$$

```
data FreeMonad f x = Var x | Op (f (FreeMonad f x))
```

The Free Monad construction

$$\_^{*} : \mathsf{TagDesc} \to \textsc{Set} \to \mathsf{TagDesc}$$
$$[E, D]^{*}\ X \mapsto [[\text{'var}\ , E], [\text{'}\Sigma\ X\ \text{'1}, D]]$$

Conclusion

We have presented...

- A rationalised universe of inductive types
- A self-encoding of the universe of types
- Some generic programming operations and constructions

Future work

- Generalising to Induction-Recursion
- Internal fixpoints
- If datatypes are data, what is design?

(Backup slides)

Desc is data!

$$\mathsf{Desc} : \mathrm{S{\scriptsize ET}}$$
$$\mathsf{Desc} \mapsto \mu\mathsf{DescD}$$

$$\text{‘1‘} : \mathsf{Desc}$$
$$\text{‘1‘} \mapsto \mathsf{con}\,[\text{'}1, []]$$

$$\text{‘}\Sigma\text{‘} : (S : \mathrm{S{\scriptsize ET}})(D : S \to \mathsf{Desc}) \to \mathsf{Desc}$$
$$\text{‘}\Sigma\text{‘}\ S\ D \mapsto \mathsf{con}\left[\text{'}\Sigma, [S, [D, []]]\right]$$

$$\text{‘ind}\times\text{‘} : (D : \mathsf{Desc}) \to \mathsf{Desc}$$
$$\text{‘ind}\times\text{‘}\ D \mapsto \mathsf{con}\,[\text{'ind}\times, [D, []]]$$

$$\text{‘hind}\times\text{‘} : (H : \mathrm{S{\scriptsize ET}})(D : \mathsf{Desc}) \to \mathsf{Desc}$$
$$\text{‘hind}\times\text{‘}\ H\ D \mapsto \mathsf{con}\left[\text{'hind}\times, [H, [D, []]]\right]$$

Skyhooks all the way up?

$$
\begin{array}{llll}
\mathsf{Desc}^n & : \textsc{Set}^{n+1} & [\![\_]\!] : \mathsf{Desc}^n \to \textsc{Set}^n \to \textsc{Set}^n & \\
\text{'}1 & : \mathsf{Desc}^n & [\![\text{'}1]\!]\, X & \mapsto 1 \\
\text{'}\Sigma\,(S : \textsc{Set}^n)\,(D : S \to \mathsf{Desc}^n) & : \mathsf{Desc}^n & [\![\text{'}\Sigma\; S\; D]\!]\, X & \mapsto (s : S) \times [\![D\; s]\!]\, X \\
\text{'ind}\times\,(D : \mathsf{Desc}^n) & : \mathsf{Desc}^n & [\![\text{'ind}\times\; D]\!]\, X & \mapsto X \times [\![D]\!]\, X \\
\text{'hind}\times\,(H : \textsc{Set}^n)\,(D : \mathsf{Desc}^n) & : \mathsf{Desc}^n & [\![\text{'hind}\times\; H\; D]\!]\, X & \mapsto (H \to X) \times [\![D]\!]\, X
\end{array}
$$

$$\mathsf{Desc}^{n+1} : \textsc{Set}^{n+2}$$

$$
\begin{array}{l}
\mathsf{DescD}^n : \mathsf{Desc}^{n+1} \\
\mathsf{Desc}^n : \textsc{Set}^{n+1} \\
\mathsf{Desc}^n \mapsto \mu\mathsf{DescD}^n
\end{array}
$$

Alternative encoding

$$
\begin{array}{ll}
\mathsf{Desc}^n & : \mathrm{SET}^{n+1} \\
\text{'}1 & : \mathsf{Desc}^n \\
\text{'}\Sigma\ (S : \mathrm{SET}^n)\ (D : S \to \mathsf{Desc}^n) & : \mathsf{Desc}^n \\
\text{'ind}\times\ (D : \mathsf{Desc}^n) & : \mathsf{Desc}^n \\
\text{'hind}\times\ (H : \mathrm{SET}^n)\ (D : \mathsf{Desc}^n) & : \mathsf{Desc}^n \\
\text{'}\sigma\ (E : \mathsf{En}^n)\ (B : \pi\ E\ \lambda\_.\ \mathsf{Desc}^n) & : \mathsf{Desc}^n \\
\text{'}\pi\ (E : \mathsf{En}^n)\ (D : \#E \to \mathsf{Desc}^n) & : \mathsf{Desc}^n
\end{array}
$$

$$
\begin{array}{ll}
[\![\_]\!] & : \mathsf{Desc}^n \to \mathrm{SET}^n \to \mathrm{SET}^n \\
[\![\text{'}1]\!]\ X & \mapsto 1 \\
[\![\text{'}\Sigma\ S\ D]\!]\ X & \mapsto (s : S) \times [\![D\ s]\!] X \\
[\![\text{'ind}\times\ D]\!]\ X & \mapsto X \times [\![D]\!]\ X \\
[\![\text{'hind}\times\ H\ D]\!]\ X & \mapsto (H \to X) \times [\![D]\!]\ X \\
[\![\text{'}\sigma\ E\ B]\!]\ X & \mapsto (e : \#E) \times [\![\mathsf{switch}\ E\ e\ (\lambda\_.\ \mathsf{Desc})\ B]\!]\ X \\
[\![\text{'}\pi\ E\ D]\!]\ X & \mapsto \pi\ E\ \lambda e.\ [\![D\ e]\!]\ X
\end{array}
$$

Universe of inductive families

$$
\begin{array}{ll}
\mathsf{IDesc}\,(I : \mathrm{SET}) & : \mathrm{SET} \\
\text{'var}\,(i : I) & : \mathsf{IDesc}\ I \\
\text{'k}\,(A : \mathrm{SET}) & : \mathsf{IDesc}\ I \\
(D : \mathsf{IDesc}\ I)\ \text{'}\!\times (D : \mathsf{IDesc}\ I) & : \mathsf{IDesc}\ I \\
\text{'}\Sigma\,(S : \mathrm{SET})\,(D : S \to \mathsf{IDesc}\ I) & : \mathsf{IDesc}\ I \\
\text{'}\Pi\,(S : \mathrm{SET})\,(D : S \to \mathsf{IDesc}\ I) & : \mathsf{IDesc}\ I
\end{array}
$$

$$
\begin{array}{lll}
[\![\_]\!] & :_{(I:\mathrm{SET}) \to} & \mathsf{IDesc}\ I \to (I \to \mathrm{SET}) \to \mathrm{SET} \\
[\![\text{'var}\ i]\!]_I\ X & \mapsto & X\ i \\
[\![\text{'k}\ K]\!]_I\ X & \mapsto & K \\
[\![D\ \text{'}\!\times D']\!]_I\ X & \mapsto & [\![D]\!]_I\ X \times [\![D']\!]_I\ X \\
[\![\text{'}\Sigma\ S\ D]\!]_I\ X & \mapsto & (s : S) \times [\![D\ s]\!]_I\ X \\
[\![\text{'}\Pi\ S\ D]\!]_I\ X & \mapsto & (s : S) \to [\![D\ s]\!]_I\ X
\end{array}
$$