The University of Nottingham

Universes for Generic Programming in Epigram

Constructing Strictly Positive Families

Peter Morris

pwm@cs.nott.ac.uk

University of Nottingham

The challenge

The University of Nottingham

To make a system for generic programming in Epigram

Strictly Positive Types

The University of Nottingham

- A large and very useful class of types
  - Built from $\mu, 0, +, 1, \times, K \rightarrow$
  - $e.g.\text{List}\ A = \mu X.1 + (A \times X)$

Strictly Positive Types

The University of Nottingham

- A large and very useful class of types
  - Built from $\mu, 0, +, 1, \times, K \rightarrow$
  - $e.g.\text{List } A = \mu X.1 + (A \times X)$

- We build a universe of strictly positive types
  - Define a type of SPT codes
  - Reflect these codes as types
  - This gives us a generic programming

SPT codes

The University of Nottingham

$$\underline{\text{data}}\ \ \frac{n\ :\ \mathsf{Nat}}{\mathsf{SPT}\ n\ :\ \star}\ \ \underline{\text{where}}$$

$$\frac{}{\mathsf{vz}\ :\ \mathsf{SPT}\ (\mathsf{s}\ n)} \qquad \frac{T\ :\ \mathsf{SPT}\ n}{\mathsf{vs}\ T\ :\ \mathsf{SPT}\ (\mathsf{s}\ n)} \qquad \frac{}{\text{`0'}\ :\ \mathsf{SPT}\ n}$$

$$\frac{S, T\ :\ \mathsf{SPT}\ n}{S\ \text{`}+\text{'}\ T\ :\ \mathsf{SPT}\ n} \qquad \frac{}{\text{`1'}\ :\ \mathsf{SPT}\ n} \qquad \frac{S, T\ :\ \mathsf{SPT}\ n}{S\ \text{`}\times\text{'}\ T\ :\ \mathsf{SPT}\ n}$$

$$\frac{K\ :\ \star \quad T\ :\ \mathsf{SPT}\ n}{K\ \text{`}\rightarrow\text{'}\ T\ :\ \mathsf{SPT}\ n} \qquad \frac{F\ :\ \mathsf{SPT}\ (\mathsf{s}\ n)}{\text{`}\mu\text{'}\ F\ :\ \mathsf{SPT}\ n}$$

Building a universe

The University of Nottingham

$$\underline{\text{data}}\ \frac{T\ :\ \mathsf{SPT}\ n \quad \vec{T}\ :\ \mathsf{Tel}\ \mathsf{SPT}\ n}{\mathsf{El}\ T\ \vec{T}\ :\ \star}\ \underline{\text{where}}\ \ \ldots$$

$$\underline{\text{data}}\ \frac{X\ :\ \mathsf{Nat}\ \rightarrow\ \star \quad n\ :\ \mathsf{Nat}}{\mathsf{Tel}\ X\ n\ :\ \star}\ \underline{\text{where}}$$

$$\frac{}{\varepsilon\ :\ \mathsf{Tel}\ X\ n} \qquad \frac{\vec{T}\ :\ \mathsf{Tel}\ X\ n \quad x\ :\ X\ n}{\vec{T} :: x\ :\ \mathsf{Tel}\ X\ (\mathsf{s}\ n)}$$

Sums and Products

The University of Nottingham

$$\frac{s \;:\; \mathsf{El}\ S\ \vec{T}}{\mathsf{inl}\ s \;:\; \mathsf{El}\ (S\ \text{`}+\text{'}\ T)\ \vec{T}} \qquad \frac{t \;:\; \mathsf{El}\ T\ \vec{T}}{\mathsf{inr}\ t \;:\; \mathsf{El}\ (S\ \text{`}+\text{'}\ T)\ \vec{T}}$$

$$\frac{}{\mathsf{void} \;:\; \mathsf{El}\ \text{`}1\text{'}\ \vec{T}} \qquad \frac{s \;:\; \mathsf{El}\ S\ \vec{T} \quad t \;:\; \mathsf{El}\ T\ \vec{T}}{\mathsf{pair}\ s\ t \;:\; \mathsf{El}\ (S\ \text{`}\times\text{'}\ T)\ \vec{T}}$$

Fixed Point, variables

The University of Nottingham

$$\frac{f \ : \ \mathsf{El}\ F\ (\vec{T} :: \text{`}\mu\text{'}\ F)}{\mathsf{in}\ f \ : \ \mathsf{El}\ (\text{`}\mu\text{'}\ F)\ \vec{T}}$$

$$\frac{e \ : \ \mathsf{El}\ T\ \vec{T}}{\mathsf{top}\ e \ : \ \mathsf{El}\ \mathsf{vz}\ (\vec{T} :: T)} \qquad \frac{e \ : \ \mathsf{El}\ T\ \vec{T}}{\mathsf{pop}\ e \ : \ \mathsf{El}\ (\mathsf{vs}\ T)\ (\vec{T} :: S)}$$

List, nil, cons

The University of Nottingham

- $\underline{\text{let}}$ ‘List’ : SPT (s z) $\Rightarrow$ ‘μ’ (‘1’‘+’(vz‘$\times$’(vs vz))

List, nil, cons

The University of Nottingham

- $\underline{\text{let}}$ ‘List’ : SPT (s z) $\Rightarrow$ ‘μ’ (‘1’ ‘$+$’ (vz ‘$\times$’ (vs vz))

- $\underline{\text{let}}$ $\dfrac{}{\text{‘}\varepsilon\text{’} : \text{El ‘List’ } [A]}$

  ‘ε’ $\Rightarrow$ in (inl void)

List, nil, cons

The University of Nottingham

- $\underline{\text{let}}\ \text{`List'} : \mathsf{SPT}\ (\mathsf{s}\ \mathsf{z}) \Rightarrow \text{`}\mu\text{'}\ (\text{`}1\text{'}\ \text{`}+\text{'}\ (\mathsf{vz}\ \text{`}\times\text{'}\ (\mathsf{vs}\ \mathsf{vz}))$
- $\underline{\text{let}}\ \dfrac{}{\text{`}\varepsilon\text{'} : \mathsf{El}\ \text{`List'}\ [A]}$

  $\text{`}\varepsilon\text{'} \Rightarrow \mathsf{in}\ (\mathsf{inl}\ \mathsf{void})$
- $\underline{\text{let}}\ \dfrac{a : \mathsf{El}\ A\ []\quad as : \mathsf{El}\ \text{`List'}\ [A]}{a\text{`::'}as : \mathsf{El}\ \text{`List'}\ [A]}$

  $a\text{`::'}as \Rightarrow \mathsf{in}\ (\mathsf{inr}\ (\mathsf{pair}\ (\mathsf{pop}\ (\mathsf{top}\ a))\ (\mathsf{top}\ as)))$

Generic Programming

The University of Nottingham

- Generic programs are parametic in an SPT code

  $$\underline{\text{let}}\ \frac{T : \mathsf{SPT}\ n \quad t : \mathsf{EI}\ T\ \vec{T}}{\mathbf{gfoo}_T\, t : T}$$

  gfoo will work for any SPT

Generic Programming

The University of Nottingham

- Generic programs are parametic in an SPT code

  $$\underline{\text{let}}\ \frac{T : \mathsf{SPT}\ n \quad t : \mathsf{El}\ T\ \vec{T}}{\mathbf{gfoo}_T\, t : T}$$

  gfoo will work for any SPT
- Can define functorial map, remove '$\rightarrow$' and we can define decidable equality

Generic Programming

The University of Nottingham

- Generic programs are parametic in an SPT code

  $$\underline{\text{let}}\ \frac{T : \mathsf{SPT}\ n \quad t : \mathsf{El}\ T\ \vec{T}}{\mathbf{gfoo}_T\, t : T}$$

  gfoo will work for any SPT
- Can define functorial map, remove '$\rightarrow$' and we can define decidable equality
- Different classes of types support different operations... so we need a number of universes like this

Inductive Families

The University of Nottingham

- There's an obvious imbalance, modelling SPTs using inductive families

Inductive Families

The University of Nottingham

- There's an obvious imbalance, modelling SPTs using inductive families
- Can we genelalise this technique to 'Strictly Positive Families'?

SPF

The University of Nottingham

$$\underline{\text{let}} \quad \frac{\vec{I} \ : \ \mathsf{Vec} \ \star \ n \quad O \ : \ \star}{\mathsf{SPF} \ \vec{I} \ O \ : \ \star} \quad \underline{\text{where}} \quad \ldots$$

SPF

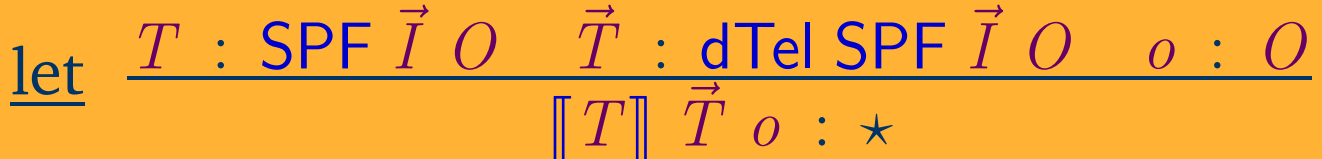
The University of Nottingham

$$\underline{\text{let}}\ \frac{\vec{I} : \mathsf{Vec}\ \star\ n \quad O : \star}{\mathsf{SPF}\ \vec{I}\ O : \star}\ \underline{\text{where}}\ \ldots$$

$$\underline{\text{data}}\ \frac{X : \mathsf{Nat} \to \star \quad n : \mathsf{Nat}}{\mathsf{Tel}\ X\ n : \star}\ \underline{\text{where}}\ \ldots$$

$$\underline{\text{let}}\ \frac{}{\text{`Tel'} : \mathsf{SPF}\ [\mathsf{Nat}]\ \mathsf{Nat}}\ \ldots$$

Interpreting SPF



Interpreting SPF

The University of Nottingham

$$\underline{\text{let}}\ \frac{T : \mathsf{SPF}\ \vec{I}\ O \quad \vec{T} : \mathsf{dTel}\ \mathsf{SPF}\ \vec{I}\ O \quad o : O}{[\![T]\!]\ \vec{T}\ o : \star}\ \underline{\text{where}}\ \ldots$$

$$[\![\text{`Tel'}]\!]\ [\text{`SPT'}] : \mathsf{Nat} \to \star$$

Interpreting SPF

$$\underline{\text{let}}\ \frac{T : \mathsf{SPF}\ \vec{I}\ O \quad \vec{T} : \mathsf{dTel}\ \mathsf{SPF}\ \vec{I}\ O \quad o : O}{[\![T]\!]\ \vec{T}\ o : \star}\ \underline{\text{where}}\ \ldots$$

$$[\![\text{`Tel'}]\!]\ [\text{`SPT'}] : \mathsf{Nat} \to \star$$

But how do we build families?

Constant families

The University of Nottingham

$$\frac{}{\text{`0'}, \text{`1'} : \mathsf{SPF}\ \vec{I}\ O}$$

Constant families

The University of Nottingham

$$\frac{}{\text{`0', `1'} : \mathsf{SPF}\ \vec{I}\ O}$$

$$\frac{}{\mathsf{void} : [\![\text{`1'}]\!]\ \vec{T}\ o}$$

Constant families

$$\frac{}{\text{`0'}, \text{`1'} \;:\; \mathsf{SPF}\ \vec{I}\ O}$$

$$\frac{}{\mathsf{void} \;:\; [\![\text{`1'}]\!]\ \vec{T}\ o}$$

No rule for `0', obviously. Just like SPT these are the basic building blocks, we now build new codes from old.

Re-indexing

The University of Nottingham

The simplest this we can do is to ‘re-index’ a family

$$\frac{f \;:\; O \to O' \quad T \;:\; \mathsf{SPT}\ \vec{I}\ O'}{\text{`}\Delta\text{'}\ f\ T \;:\; \mathsf{SPF}\ \vec{I}\ O}$$

Re-indexing

The University of Nottingham

The simplest this we can do is to ‘re-index’ a family

$$\frac{f \;:\; O \to O' \quad T \;:\; \mathsf{SPT}\ \vec{I}\ O'}{\text{`}\Delta\text{'}\ f\ T \;:\; \mathsf{SPF}\ \vec{I}\ O}$$

$$\frac{v \;:\; [\![T]\!]\ \vec{T}\ (f\ o)}{\delta v \;:\; [\![\text{`}\Delta\text{'}\ f\ T]\!]\ \vec{T}\ o}$$

Re-indexing

The University of Nottingham

The simplest this we can do is to ‘re-index’ a family

$$\frac{f \ : \ O \to O' \quad T \ : \ \mathsf{SPT} \ \vec{I} \ O'}{\text{`}\Delta\text{'} \ f \ T \ : \ \mathsf{SPF} \ \vec{I} \ O}$$

$$\frac{v \ : \ [\![T]\!] \ \vec{T} \ (f \ o)}{\delta v \ : \ [\![\text{`}\Delta\text{'} \ f \ T]\!] \ \vec{T} \ o}$$

What happens if $f \ : \ O' \to O$?

Two things

Two things

$$\frac{f \;:\; O' \to O \qquad T \;:\; \mathsf{SPT}\ \vec{I}\ O'}{\text{`}\Sigma\text{'}\ f\ T \qquad\qquad :\ \mathsf{SPF}\ \vec{I}\ O}$$

We can pick an $o' \;:\; O'$ s.t. $f\ o' = o$

$$\frac{v \;:\; \exists o' \;:\; O' \Rightarrow f\ o' = o \to [\![T]\!]\ \vec{T}\ o'}{\sigma v \;:\; [\![\text{`}\Sigma\text{'}\ f\ T]\!]\ \vec{T}\ o}$$

Two things

$$\frac{f\ :\ O' \to O \quad T\ :\ \mathsf{SPT}\ \vec{I}\ O'}{\text{`}\Sigma\text{'}\ f\ T, \text{`}\Pi\text{'}\ f\ T\ :\ \mathsf{SPF}\ \vec{I}\ O}$$

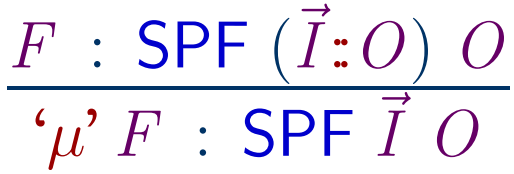
We can pick an $o'\ :\ O'$ s.t. $f\ o'{=}o$

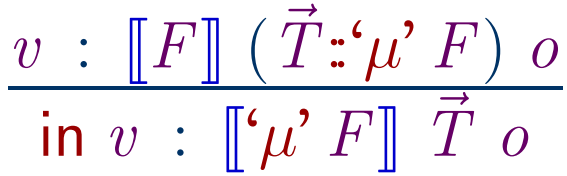
$$\frac{v\ :\ \exists o'\ :\ O' \Rightarrow f\ o'{=}o \to [\![T]\!]\ \vec{T}\ o'}{\sigma v\ :\ [\![\text{`}\Sigma\text{'}\ f\ T]\!]\ \vec{T}\ o}$$

Or we can consider *all* such values

$$\frac{v\ :\ \forall o'\ :\ O' \Rightarrow f\ o'{=}o \to [\![T]\!]\ \vec{T}\ o'}{\pi v\ :\ [\![\text{`}\Pi\text{'}\ f\ T]\!]\ \vec{T}\ o}$$

Fixed Points, variables





Fixed Points, variables

The University of Nottingham

$$\frac{F \;:\; \mathsf{SPF}\ (\vec{I} :: O)\ O}{\text{`}\mu\text{'}\ F \;:\; \mathsf{SPF}\ \vec{I}\ O}$$

$$\frac{v \;:\; [\![F]\!]\ (\vec{T} :: \text{`}\mu\text{'}\ F)\ o}{\mathsf{in}\ v \;:\; [\![\text{`}\mu\text{'}\ F]\!]\ \vec{T}\ o}$$

$$\frac{}{\mathsf{vz} \;:\; \mathsf{SPF}\ (\vec{I} :: I)\ I}$$

$$\frac{v \;:\; [\![T]\!]\ \vec{T}\ o}{\mathsf{top}\ v \;:\; [\![\mathsf{vz}]\!]\ (\vec{T} :: T)\ o}$$

$$\frac{T \;:\; \mathsf{SPF}\ \vec{I}\ O}{\mathsf{vs}\ T \;:\; \mathsf{SPF}\ (\vec{I} :: I)\ O}$$

$$\frac{v \;:\; [\![T]\!]\ \vec{T}\ o}{\mathsf{pop}\ v \;:\; [\![\mathsf{vs}\ T]\!]\ (\vec{T} :: S)\ o}$$

One last code

The University of Nottingham

So far there's no way to combine SPFs, how to do that?

One last code

The University of Nottingham

So far there's no way to combine SPFs, how to do that? Finite choice between constructors, based on part of the index:

$$\frac{f \ : \ \forall t \ : \ \mathsf{Fin}\ n \Rightarrow \mathsf{SPF}\ \vec{I}\ O}{\text{`Tag'}\ f \ : \ \mathsf{SPF}\ \vec{I}\ (O \times \mathsf{Fin}\ n)}$$

$$\frac{v \ : \ [\![f \ t]\!]\ \vec{T}\ o}{\mathsf{tag}\ v \ : \ [\![\text{`Tag'} f]\!]\ \vec{T}\ (o; t)}$$

defining Sum and Product

The University of Nottingham

$$\underline{\text{let}}\ \frac{A, B\ :\ \mathsf{SPF}\ \vec{I}\ O}{A\ \text{`}+\text{'}\ B\ :\ \mathsf{SPF}\ \vec{I}\ O}$$

$$A\ \text{`}+\text{'}\ B\ \Rightarrow\ \text{`}\Sigma\text{'}\mathbf{fst}\ \left(\text{`Tag'}\ \left(\lambda\ \begin{array}{lcl}\mathsf{fz} & \Rightarrow & A\\ \mathsf{fs}\ \mathsf{fz} & \Rightarrow & B\end{array}\right)\right)$$

$$\underline{\text{let}}\ \frac{A, B\ :\ \mathsf{SPF}\ \vec{I}\ O}{A\ \text{`}\times\text{'}\ B\ :\ \mathsf{SPF}\ \vec{I}\ O}$$

$$A\ \text{`}\times\text{'}\ B\ \Rightarrow\ \text{`}\Pi\text{'}\mathbf{fst}\ \left(\text{`Tag'}\ \left(\lambda\ \begin{array}{lcl}\mathsf{fz} & \Rightarrow & A\\ \mathsf{fs}\ \mathsf{fz} & \Rightarrow & B\end{array}\right)\right)$$

Generic Epigram

The University of Nottingham

- All Epigram data types have a code in this universe

$$\text{`Tel'} \Rightarrow \text{`}\mu\text{'} \left(\begin{array}{l} (\text{`}\Sigma\text{'}\ (\mathbf{const}\ \mathsf{z})\ \text{`}1\text{'}) \\ \quad \text{`}+\text{'}\ (\text{`}\Sigma\text{'}\mathsf{s}\ (\mathsf{vz}\ \text{`}\times\text{'}\ \mathsf{vs}\ \mathsf{vz})) \end{array} \right)$$

Generic Epigram

The University of Nottingham

- All Epigram data types have a code in this universe

$$\text{`Tel'} \Rightarrow \text{`}\mu\text{'} \left(\begin{array}{l} (\text{`}\Sigma\text{'}\ (\mathbf{const}\ \mathsf{z})\ \text{`}1\text{'}) \\ \quad \text{`}+\text{'}\ (\text{`}\Sigma\text{'}\mathsf{s}\ (\mathsf{vz}\ \text{`}\times\text{'}\ \mathsf{vs}\ \mathsf{vz})) \end{array} \right)$$

- This SPF universe gives us generic programming for inductive families, we can define functorial map, equality (with restricted 'Π') etc...

The End

The University of Nottingham

- Just as strictly positive types are made from $\mu, 0, +, 1, \times$ and $K \to$, strictly positive families are constricted from $\mu, 0, 1, \Sigma, \Delta, \Pi$ and Tag

The End

The University of Nottingham

- Just as strictly positive types are made from $\mu, 0, +, 1, \times$ and $K\ \rightarrow$, strictly positive families are constricted from $\mu, 0, 1, \Sigma, \Delta, \Pi$ and Tag

- The question is how to create a usable generic programming system for Epigram using these codes, in the pipeline

The End

The University of Nottingham

- Just as strictly positive types are made from $\mu, 0, +, 1, \times$ and $K\ \rightarrow$, strictly positive families are constricted from $\mu, 0, 1, \Sigma, \Delta, \Pi$ and Tag
- The question is how to create a usable generic programming system for Epigram using these codes, in the pipeline
- More information: “Exploring the Regular Tree Types”, “Constructing Strictly Positive Families” and my thesis, soon.

SPF

The University of Nottingham

$$\underline{\text{data}}\ \frac{\vec{I} : \mathsf{Vec}\ \star\ n \quad O : \star}{\mathsf{SPF}\ \vec{I}\ O : \star}\ \underline{\text{where}}$$

$$\frac{}{\mathsf{vz} : \mathsf{SPF}\ (\vec{I} :: O)\ O} \qquad \frac{T : \mathsf{SPF}\ \vec{I}\ O}{\mathsf{vs}\ T : \mathsf{SPF}\ (\vec{I} :: I)\ O}$$

$$\frac{f : \forall t : \mathsf{Fin}\ n \Rightarrow \mathsf{SPF}\ \vec{I}\ O}{\text{`Tag'}\ f : \mathsf{SPF}\ \vec{I}\ (O \times \mathsf{Fin}\ n)} \qquad \frac{}{\text{`0'}, \text{`1'} : \mathsf{SPF}\ \vec{I}\ O}$$

$$\frac{f : O \to O' \quad T : \mathsf{SPF}\ \vec{I}\ O}{\text{`}\Sigma\text{'}O\ f\ T : \mathsf{SPF}\ \vec{I}\ O'} \qquad \frac{f : O' \to O \quad T : \mathsf{SPF}\ \vec{I}\ O}{\text{`}\Delta\text{'}O\ f\ T : \mathsf{SPF}\ \vec{I}\ O'}$$

$$\frac{f : O \to O' \quad T : \mathsf{SPF}\ \vec{I}\ O}{\text{`}\Pi\text{'}O\ f\ T : \mathsf{SPF}\ \vec{I}\ O'} \qquad \frac{T : \mathsf{SPF}\ (\vec{I} :: O)\ O}{\text{`}\mu\text{'}\ T : \mathsf{SPF}\ \vec{I}\ O}$$

El

$$\underline{\text{data}}\ \frac{T : \mathsf{F}\ \vec{I}\ O \quad \vec{T} : \mathsf{Tel}\ \vec{I} \quad o : O}{[\![T]\!]\,\vec{T}\ o : \star}\ \underline{\text{where}}$$

$$\frac{v : [\![T]\!]\,\vec{T}\ \vec{X}\ o}{\mathsf{top}\ v : [\![\mathsf{vz}]\!](\vec{T} :: T)\ o} \qquad \frac{v : [\![T]\!]\,\vec{T}\ \vec{X}\ o}{\mathsf{pop}\ v : [\![\mathsf{vs}\ T]\!](\vec{T} :: T)\ o}$$

$$\frac{v : [\![f\ t]\!]\,\vec{T}\ o}{\mathsf{tag}\ v : [\![\text{`Tag'}f]\!]\,\vec{T}\ (o; t)} \qquad \frac{}{\mathsf{void} : [\![\text{`1'}]\!]\,\vec{T}\ o}$$

$$\frac{v : [\![T]\!]\,\vec{T}\ o}{\sigma\, o\, v : [\![\text{`}\Sigma\text{'}f\ T]\!]\,\vec{T}\ (f\ o)} \qquad \frac{v : [\![T]\!]\,\vec{T}\ (f\ o)}{\delta v : [\![\text{`}\Delta\text{'}f\ T]\!]\,\vec{T}\ o}$$

$$\frac{\vec{v} : \forall o : O;\ p : (f\ o) = o'[\![T]\!]\,\vec{T}\ o}{\pi\vec{v} : [\![\text{`}\Pi\text{'}f\ T]\!]\,\vec{T}\ o'} \qquad \frac{v : [\![T]\!](\vec{T} :: (\text{`}\mu\text{'}\ T))\ o}{\mathsf{in}\ v : [\![\text{`}\mu\text{'}\ T]\!]\,\vec{T}\ o}$$