The Design Invariant

If $\cdot ; \cdot \vdash f : \{E\} \longrightarrow \{E\}$

then f is a monotone function

Hence:

$$\cdot ; \cdot \vdash \mathrm{fix}\ x.\ f(x) : \{E\}$$

is a well-defined fixed point

Datafun: A Language for Fixed-Point Computations

Neel Krishnaswami
University of Cambridge

FLOLAC 2024
Taipei, Taiwan

Syntax

Types $\quad A, B ::= 1 \mid A \times B \mid A \to B \mid A + B \mid \{E\} \mid \Box A$

Eq types $\quad E, F ::= 1 \mid E + F \mid E \times F \mid \{E\}$

$\Gamma ::= \cdot \mid \Gamma, x : A \qquad \Delta ::= \cdot \mid \Delta, x : A$

$$\begin{aligned} e ::= {} & () \mid (e, e') \mid \pi_i(e) \mid \lambda x . e \mid e\, e' \mid \mathrm{in}_i(e) \\ & \mid \mathrm{case}\,(e, \overrightarrow{\mathrm{in}_i\, x_i \to e_i}) \mid \emptyset \mid e \cup e' \mid \mathrm{for}\ x \in e .\ e' \mid \{e\} \\ & \mid \mathrm{box}(e) \mid \mathrm{let}\ \mathrm{box}(x) = e\ \mathrm{in}\ e' \\ & \mid \mathrm{fix}\ x : L . e \mid x \mid \underline{x} \mid e_1 = e_2 \mid \mathrm{empty?}(e) \end{aligned}$$

$$\boxed{\Delta ; \Gamma \vdash e : A}$$

Typing

$$\frac{x : A \in \Gamma}{\Delta ; \Gamma \vdash x : A} \qquad \frac{}{\Delta ; \Gamma \vdash () : 1} \qquad \frac{\Delta ; \Gamma \vdash e_1 : A_1 \quad \Delta ; \Gamma \vdash e_2 : A_2}{\Delta ; \Gamma \vdash (e_1, e_2) : A_1 \times A_2}$$

$$\frac{\Delta ; \Gamma \vdash e : A_1 \times A_2}{\Delta ; \Gamma \vdash \pi_i(e) : A_i} \qquad \frac{\Delta ; \Gamma, x : A \vdash e : B}{\Delta ; \Gamma \vdash \lambda x . e : A \to B} \qquad \frac{\Delta ; \Gamma \vdash e_1 : A \to B \quad \Delta ; \Gamma \vdash e_2 : A}{\Delta ; \Gamma \vdash e_1\, e_2 : B}$$

$$\frac{\Delta ; \Gamma \vdash e : A_i}{\Delta ; \Gamma \vdash in_i(e) : A_1 + A_2} \qquad \frac{\Delta ; \Gamma \vdash e : A_1 + A_2 \quad \Delta ; \Gamma, x_1 : A_1 \vdash e_1 : B \quad \Delta ; \Gamma, x_2 : A_2 \vdash e_2 : B}{\Delta ; \Gamma \vdash \mathrm{case}(e, in_1\, x_1 \to e_1, in_2\, x_2 \to e_2) : B}$$

Modal Typing

$$\frac{x : A \in \Delta}{\Delta ; \Gamma \vdash \underline{x} : A}$$

$$\frac{\Delta ; \cdot \vdash e : A}{\Delta ; \Gamma \vdash \mathrm{box}(e) : \Box A}$$

$$\cfrac{\cfrac{\Delta ; x : A \vdash x : A}{\Delta ; \cdot \vdash \lambda x . x : A \to A}}{\Delta ; \Gamma \vdash \mathrm{box}(\lambda x . x) : \Box(A \to A)}$$

$$\frac{\Delta ; \Gamma \vdash e_1 : \Box A \qquad \Delta , x : A ; \Gamma \vdash e_2 : B}{\Delta ; \Gamma \vdash \mathrm{let\ box}(x) = e_1 \ \mathrm{in}\ e_2 : B}$$

Data Structures

$$E, F ::= 1 \mid E+F \mid E \times F \mid \{E\}$$

$$\frac{\Delta; \cdot \vdash e : E}{\Delta; \Gamma \vdash \{e\} : \{E\}}$$

$$\frac{}{\Delta; \Gamma \vdash \emptyset : \{E\}}$$

$$\frac{\Delta; x : \{E\} \vdash e : \{E\}}{\Delta; \Gamma \vdash \mathrm{fix}\ x.\, e : \{E\}}$$

$$\frac{\Delta; \Gamma \vdash e_1 : \{E\} \qquad \Delta; L \vdash e_2 : \{E\}}{\Delta; \Gamma \vdash e_1 \cup e_2 : \{E\}}$$

$$\frac{\Delta; \Gamma \vdash e_1 : \{E\} \qquad \Delta, x : E; \Gamma \vdash e_2 : \{F\}}{\Delta; \Gamma \vdash \mathrm{for}\ x \in e_1 .\, e_2 : \{F\}}$$

$$\frac{\Delta; \cdot \vdash e_1 : E \qquad \Delta; \cdot \vdash e_2 : E}{\Delta; \Gamma \vdash e_1 = e_2 : \mathrm{bool}}$$

$$\frac{\Delta; \cdot \vdash e : \{1\}}{\Delta; \cdot \vdash \mathrm{empty?}(e) : \mathrm{bool}}$$

$1+1$

$$\{a, b, c\} \subseteq \{a, b, c, d\}$$

$$\{\{a, b\}, \{c\}\} \stackrel{?}{\subseteq} \{\{a, b, c\}, \{c\}\}$$

The Design Invariant

If $\cdot\,;\cdot \vdash f : \{E\} \longrightarrow \{E\}$

then f is a monotone function

The Design Invariant

If $\cdot\,;\,\cdot \vdash f : \{E\} \longrightarrow \{E\}$

then f is a monotone function

Hence:

$$\cdot\,;\,\cdot \vdash \mathrm{fix}\ x.\ f(x) : \{E\}$$

is a well-defined fixed point

$$\boxed{\text{Typechecking ensures monotonicity}}$$

$$x : \{E\} \vdash e : \{E\}$$

$$[\emptyset / x]\, e \subseteq [\{\varsigma\} / x]\, e$$

What's In the $\Box$?

- Intuitively, $\Box A$ represents values which <u>don't</u> <u>change</u>.
- Hence, they cannot vary over during a fixed-point computation
- Details in the next session!

Programming in Datafun

$$\mathrm{map} : \square(E \to F) \to \{E\} \to \{F\}$$

$$\mathrm{map}\ \mathrm{box}(f)\ es =$$

$$\quad \mathrm{for}\ x \in es.\ \ \{\underline{f}\ \underline{x}\}$$

Programming in Datafun

$$\text{map} : \Box(E \to F) \to \{E\} \to \{F\}$$

$$\text{map}\ \text{box}(f)\ es =$$
$$\quad \text{for } x \in es.\ \{\underline{f}\ \underline{x}\}$$

$$\text{badmap} : (E \to F) \to \{E\} \to \{F\}$$

$$\text{badmap}\ f\ es =$$
$$\quad \text{for } x \in es.\ \{f\ \underline{x}\}$$

not in scope here!

Programming in Datafun

$$\mathrm{filter} : (E \to 1+1) \to \{E\} \to \{E\}$$

$$\begin{aligned}
&\mathrm{filter}\ f\ es = \\
&\quad \mathrm{for}\ x \in es.\ \mathrm{case}(f(\underline{x}), \\
&\qquad\qquad \mathrm{in}_1\,() \to \{x\}, \\
&\qquad\qquad \mathrm{in}_2\,() \to \emptyset)
\end{aligned}$$

Programming in Datafun

$$\text{filter} : (E \to 1+1) \to \{E\} \to \{E\}$$

$$\begin{aligned}
&\text{filter } f \text{ es} = \\
&\qquad \text{for } x \in \text{es}. \quad \text{if } f(\underline{x}) \text{ then} \\
&\qquad\qquad\qquad\qquad \{x\} \\
&\qquad\qquad\qquad \text{else} \\
&\qquad\qquad\qquad\qquad \emptyset
\end{aligned}$$

Programming in Datafun

$$\text{prod} : \{E\} \to \{F\} \to \{E \times F\}$$

$$\text{prod}\ xs\ ys =$$

$$\quad \text{for } x \in xs.$$

$$\quad\quad \text{for } y \in ys.\ \ \{(\underline{x}, \underline{y})\}$$

Programming in Datafun

$$\text{member} : \square E \to \{E\} \to \text{bool}$$

$$\text{member}\ \text{box}(x)\ ys =$$

$$\text{let}\ b : \{1\} = \left[\begin{array}{l} \text{for}\ y \in ys.\ \text{if}\ \underline{x} = \underline{y}\ \text{then} \\ \qquad \{()\} \\ \quad \text{else} \\ \qquad \emptyset \end{array}\right]$$

$$\text{in}$$

$$\text{empty?}(b)$$

Programming in Datafun

intersect : $\{E\} \to \{E\} \to \{E\}$

intersect xs ys =

  for $x \in xs$.

    if member box($\underline{x}$) ys then

      $\{x\}$

    else

      $\emptyset$

Programming in Datafun

$$\text{compose} : \{E \times F\} \to \{F \times G\} \to \{E \times G\}$$

$$\text{compose} \quad R \quad S \quad =$$

$$\quad \text{for } (e, f) \in R.$$

$$\qquad \text{for } (f', g) \in S.$$

$$\qquad\quad \text{if } \underline{f} = \underline{f}' \text{ then}$$

$$\qquad\qquad \{(\underline{e}, \underline{g})\}$$

$$\qquad\quad \text{else}$$

$$\qquad\qquad \emptyset$$

$$\boxed{\text{This is impossible in Datalog!}}$$

Programming in Datafun

$$\mathrm{compose} : \{E \times F\} \to \{F \times G\} \to \{E \times G\}$$

$$\mathrm{compose}\ R\ S =$$

$$\quad \mathrm{for}\ (e, f) \in R.$$

$$\qquad \mathrm{for}\ (f', g) \in S.$$

$$\qquad\quad \mathrm{case}(\ \underline{f} = \underline{f'},$$

$$\qquad\qquad in_1\ \_ \to \{(\underline{e}, \underline{g})\}$$

$$\qquad\qquad in_2\ \_ \to \emptyset\ )$$

Programming in Datafun

$$\text{compose} : \{E \times F\} \to \{F \times G\} \to \{E \times G\}$$

$$
\begin{aligned}
&\text{compose} \quad R \quad S \quad = \\
&\quad \text{for} \quad ef \in R. \\
&\quad\quad \text{for} \quad fg \in S. \\
&\quad\quad\quad \text{case}(\pi_2(e\underline{f}) = \pi_1(\underline{f}g) \\
&\quad\quad\quad\quad in_1\, \_ \to \{(\pi_1(\underline{e}f), \pi_2(f\underline{g}))\} \\
&\quad\quad\quad\quad in_2\, \_ \to \emptyset)
\end{aligned}
$$

Programming in Datafun

$$\text{compose} : \{E \times F\} \to \{F \times G\} \to \{E \times G\}$$

$$\text{compose} \quad R \quad S \quad =$$

$$\quad \text{for} \quad ef \in R.$$

$$\quad\quad \text{for} \quad fg \in S.$$

$$\quad\quad\quad \text{case}(\pi_2(\underline{ef}) = \pi_1(\underline{fg})$$

$$\quad\quad\quad\quad in_1\,\_ \to \{(\pi_1(\underline{ef}), \pi_2(\underline{fg}))\}$$

$$\quad\quad\quad\quad in_2\,\_ \to \phi)$$

Programming in Datafun

$$\text{trans} : \square\{E \times E\} \to \{E \times E\}$$

$$\text{trans box}(E) =$$

$$\quad \text{fix } R.\ \underline{E} \cup \text{compose } \underline{E}\ R$$

Regular Expressions

Assume we have

$\text{len}: \square\,\text{string} \to \mathbb{N}$

$\text{len}\ \text{box}(s)$ = "the length of s"

$\text{chars}: \square\,\text{Str} \to \{\text{Char} \times \mathbb{N}\}$

$\text{chars}\ \text{box}(s)$ = "$\{(c, i) \mid$ i-th char of s is c$\}$"

Regular Expressions

type regex = $\square$string $\to \{\mathbb{N} \times \mathbb{N}\}$

char : $\square$char $\to$ regex

char box(c) s =

for $(i, c') \in$ chars(s).

if $\underline{c} = \underline{c}'$ then

$\{(i, i+1)\}$

else

$\emptyset$

$(\underline{a} \cdot \underline{b})$ "abc abc"

Regular Expressions

$$\text{type regex} = \square\, \text{string} \to \{\mathbb{N} \times \mathbb{N}\}$$

$$\text{char} : \square\, \text{char} \to \text{regex}$$

$$\text{char box}(c)\quad s =$$

$\quad$ for $(i, c') \in \text{chars}(s)$.

$\qquad$ if $\underline{c} = \underline{c'}$ then

$\qquad\quad \{(i, i+1)\}$

$\qquad$ else

$\qquad\quad \emptyset$

Regular Expressions

$$\text{type regex} = [\,]\ \text{string} \to \{\mathbb{N} \times \mathbb{N}\}$$

$$\text{seq} : \text{regex} \to \text{regex} \to \text{regex}$$

$$\text{seq}\ R_1\ R_2\ s = \text{compose}\ (R_1\ s)\ (R_2\ s)$$

$(7, 13)$ $(13, 20)$

$(7, 20)$

Regular Expressions

$$\text{type regex} = \square\, \text{string} \to \{\mathbb{N} \times \mathbb{N}\}$$

$$\text{nil} : \text{regex}$$

$$\text{nil} \mid \text{box}(s) = \text{for } (c, i) \in \text{chars}(\text{box}(s)).\ \{(\underline{i}, \underline{i})\} \cup \{(\text{len}(\text{box}(\underline{s})), \text{len}(\text{box}(s)))\}$$

Regular Expressions

type regex = $\square$ string $\to \{\mathbb{N} \times \mathbb{N}\}$

alt : regex $\to$ regex $\to$ regex

alt R_1 R_2 s = $(R_1\, s) \cup (R_2\, s)$

Regular Expressions

$$\text{type regex} = \Box\,\text{string} \to \{\mathbb{N} \times \mathbb{N}\}$$

$$\text{star} : \Box\,\text{regex} \to \text{regex}$$

$$\text{star box}(r)\ \text{box}(s) =$$

$$\text{nil}(\text{box}(\underline{s})) \cup \text{trans box}(\underline{r}\ \text{box}(\underline{s}))$$