Typage
Master 2: Languages et Programmation

Delia Kesner, Giovanni Bernardi

2017 - 18

IRIF

université PARIS DIDEROT PARIS 7

Info

- `https://www.irif.fr/~gio/index.xhtml` web-page
- `gio (vous-savez-quoi) irif.fr` e-mail adress

  subject: “ **[typage]** … ”

- Questions? office 4026
- Stop me at 11:30 !!!! il y a le TP!!

Schedule

Check on-line

?? 21 Mars ??

Aim

introduction to

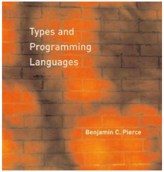
$\boxed{\text{coinduction}}$

with an application to

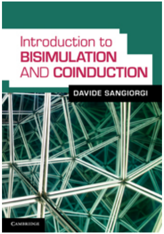
$\boxed{\text{recursive types}}$

Material

- Chapter 21 “Types and Programming Languages”



- Chapter 2 “Introduction to Bisimulation and coinduction”

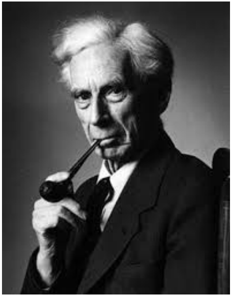


This lecture

1. Mini historical remarks
2. General motivation: circularity
3. Pot-pourri of technicalities
4. Trees and type equivalence

Who conceived types?

Who brought types into "PL"?



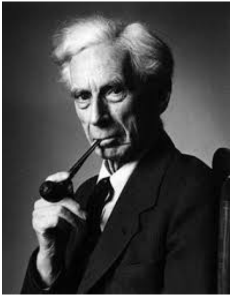
Who conceived types?

Mathematical Logic
as Based on the Theory of Types

Bertrand Russell, 1908

Why? $A \triangleq \{ x \mid x \notin x \}$

Who brought types into "PL"?



Who conceived types?

Mathematical Logic
as Based on the Theory of Types

Bertrand Russell, 1908

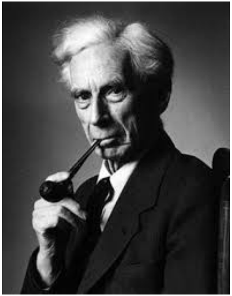
Why? $\quad A \triangleq \{\, x \mid x \notin x \,\}$

Who brought types into "PL"?

A Formulation of the
Simple Theory of Types

Alonzo Church, 1940

Why ???



Who conceived types?

Mathematical Logic
as Based on the Theory of Types

Bertrand Russell, 1908

Why? $A \triangleq \{\, x \mid x \notin x \,\}$

annus mirabilis CS

$\boxed{1936}$

Types are older than CS

Non trivial phenomenon

$$\begin{array}{lcl}
\mathit{fact} & \triangleq & \lambda x.\ \texttt{if}\ x = 0\ \texttt{then}\ 1\ \texttt{else}\ x * (\mathit{fact}(x-1)) \\
\texttt{List}\ 'a & \triangleq & [\,]\ |\ 'a : \texttt{List}\ 'a \\
M, N & ::= & x \mid \lambda x.M \mid MN
\end{array}$$

Non trivial phenomenon

$$\underline{\mathit{fact}} \quad \stackrel{\Delta}{=} \quad \lambda x.\ \texttt{if}\ x = 0\ \texttt{then}\ 1\ \texttt{else}\ x * (\underline{\mathit{fact}}(x-1))$$

$$\underline{\texttt{List}\ {}'a} \quad \stackrel{\Delta}{=} \quad [\,]\ \mid\ {}'a : \underline{\texttt{List}\ {}'a}$$

$$\underline{M, N} \quad ::= \quad x \mid \lambda x.\underline{M} \mid \underline{MN}$$

Non trivial phenomenon

$$\underline{\mathit{fact}} \quad \triangleq \quad \lambda x.\ \texttt{if}\ x = 0\ \texttt{then}\ 1\ \texttt{else}\ x * (\underline{\mathit{fact}}(x-1))$$

$$\underline{\texttt{List}\ 'a} \quad \triangleq \quad [\,] \mid\ 'a : \underline{\texttt{List}\ 'a}$$

$$\underline{M, N} \quad ::= \quad x \mid \lambda x.\underline{M} \mid \underline{MN}$$

How to treat with circularity?

structure

$$\boxed{x = F(x)}$$

x fixed point of F

least fixed points	induction	recursion
greatest fixed points	coinduction	corecursion

Induction limitation

Consider the following circular definitions — Ocaml snippet

```
# let rec a = 2::a;;
val a :  int list = [2; <cycle>]
# let rec b = (1+1)::b;;
val b :  int list = [2; <cycle>]
```

intuition

$\boxed{\texttt{a = b}}$

- How to prove `a = b` ?

  hint where is the base case in the definitions ?

- How to define `=` over lists ?

Induction hunder the hood: set-theoretic appoach

Theorem (Kleene, 1936)

Let $\langle P, \leq \rangle$ CPO and $f : P \to P$ a monotone continuous function. We have $\mu f = \bigcup_{n \geq 0} f^n(\bot)$. $\square$

Induction hunder the hood: set-theoretic appoach

A poset $\langle D, \leq \rangle$ is

- *directed* if $D \neq \emptyset$ and $\forall a, b \in D.\, \exists c \in D.\, a \leq c$ and $b \leq c$.
- a *complete partial order* (CPO) if
  - P has a bottom $\perp$ element
  - $\bigsqcup D$ exists for every directed subset of D of P

If $\langle P, \leq \rangle$, $\langle Q, \sqsubseteq \rangle$ CPO, a function $f : P \to Q$ is *continuous* if for every directed subset D of P

- $f(D)$ is directed
- $f(\bigsqcup D) = \bigsqcup f(D)$

Theorem (Kleene, 1936)

Let $\langle P, \leq \rangle$ CPO and $f : P \to P$ a monotone continuous function. We have $\mu f = \bigcup_{n \geq 0} f^n(\perp)$. $\square$

Induction hunder the hood: set-theoretic appoach

A poset $\langle D, \leq \rangle$ is

- *directed* if $D \neq \emptyset$ and D contains an upper bound c of $\{a, b\}$.
- a *complete partial order* (CPO) if
  - P has a bottom $\perp$ element
  - $\bigsqcup D$ exists for every directed subset of D of P

If $\langle P, \leq \rangle$, $\langle Q, \sqsubseteq \rangle$ CPO, a function $f : P \to Q$ is *continuous* if for every directed subset D of P

- $f(D)$ is directed
- $f(\bigsqcup D) = \bigsqcup f(D)$

Theorem (Kleene, 1936)

Let $\langle P, \leq \rangle$ CPO and $f : P \to P$ a monotone continuous function. We have $\mu f = \bigcup_{n \geq 0} f^n(\perp)$. $\square$

Induction hunder the hood: set-theoretic appoach

Typial CPO: powerset

$$S = \{a, b, c\}$$

$$\begin{tikzcd}
& S \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \\
\{a, b\} \arrow[d, dash] \arrow[dr, dash] & \{a, c\} \arrow[dl, dash] \arrow[dr, dash] & \{b, c\} \arrow[dl, dash] \arrow[d, dash] \\
\{a\} \arrow[dr, dash] & \{b\} \arrow[d, dash] & \{c\} \arrow[dl, dash] \\
& \emptyset &
\end{tikzcd}$$

Hasse diagram of set inclusion.

Theorem (Kleene, 1936)

Let $\langle P, \leq \rangle$ *CPO and* $f : P \to P$ *a monotone continuous function. We have* $\mu f = \bigcup_{n \geq 0} f^n(\bot)$*.* $\square$

Factorial as least fixed point

$$F \triangleq \lambda \underline{y}.\lambda x.\ \texttt{if}\ x = 0\ \texttt{then}\ 1\ \texttt{else}\ x * (\underline{y}(x-1))$$
$$F : (\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N})$$

- $\langle \mathbb{N} \to \mathbb{N}, \leq \rangle$ CPO with bottom $\emptyset$ and $F(y)$ continuous in y,

$$\mu y.F(y) = \bigcup_{n \geq 0} F^n(\emptyset)$$

- NB: $\mu y.F(y)$ is a function!

Factorial as least fixed point

$$
\begin{array}{lcl}
F & \triangleq & \lambda \underline{y}.\lambda x.\ \texttt{if}\ x = 0\ \texttt{then}\ 1\ \texttt{else}\ x * (\underline{y}(x-1)) \\
F & : & (\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N}) \\
\mathit{fact} & \triangleq & \mu y.F(y)
\end{array}
$$

- $\langle \mathbb{N} \to \mathbb{N}, \leq \rangle$ CPO with bottom $\emptyset$ and $F(y)$ continuous in y,

$$\mu y.F(y) = \bigcup_{n \geq 0} F^n(\emptyset)$$

- NB: $\mu y.F(y)$ is a function!

from "definition" to property

$$\boxed{\mathit{fact}(x) = \ \texttt{if}\ x = 0\ \texttt{then}\ 1\ \texttt{else}\ x * (\mathit{fact}(x-1))}$$

Induction

- Not as powerful as you may think
  - fit to define / reason on finite structures
  - we need to define / reason on circular structures too

- Set theoretically not as straightforward as it seems
  - CPO, continuous endofunctions, ...
  - other approaches to least fixed points ???

Least fixed point λ-theoretic approach

fixed-point combinator

$$\boxed{\mathcal{Y} \triangleq \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))}$$

Theorem (Kleene, 1936, "λ-definability and recursiveness")
For every λ-term M we have $\mathcal{Y}M \overset{\beta}{=} M(\mathcal{Y}M)$. □

Theorem (Morris, 1968, PhD thesis Corollary 7(a))
For every λ-term M, A if $A \overset{\beta}{=} MA$ then $\mathcal{Y}M \leq A$. □

Factorial?

- $F \triangleq \lambda y.\lambda x.\ \texttt{if}\ x = 0\ \texttt{then}\ 1\ \texttt{else}\ x * (y(x-1))$
- $\mathcal{Y}F$ is a fixed point of F
- $\mathcal{Y}F$ is the least fixed point of F, $\mathit{fact} \triangleq \mathcal{Y}F$

Can $\mathcal{Y}$ be typed ? intuitive argument

$$\text{Let } M = \lambda x.f(xx),\ \mathcal{Y} = \lambda f.MM,$$
$$\Gamma = \{x : A, x : A \to B, f : B \to C\}.$$

type derivation of sub-term of $\mathcal{Y}$

$$\boxed{\dfrac{\dfrac{}{\Gamma \vdash f : B \to C} \quad \dfrac{\dfrac{}{\Gamma \vdash x : A \to B} \quad \dfrac{}{\Gamma \vdash x : A}}{\Gamma \vdash xx : B}}{\dfrac{\Gamma \vdash f(xx) : C}{\Gamma \vdash \lambda x.f(xx) : A \to C}}}$$

We need a type that satisfies

$$\boxed{A = A \to B}$$

Recursive types

$$A \quad ::= \quad \mathcal{T} \mid \underline{x} \mid \underline{\mu x.A} \mid A \times A \mid A \to A$$

- $\mu x.T$ binds x in T, free and bound variables as expected
- μ-types are closed and contractive terms

A *contractive* if for any subexpression of A of the form

$$\mu x.\mu x_1.\mu x_2.\dots\mu x_n.B$$

the term B is not x.

- not contractive: $\mu x.x$
- contractive: $\mu x.y$ but not closed
- not contractive: $int \to \mu x.x$
- contractive: $\mu x.x \to x$

Recursive types

$$A \quad ::= \quad \mathcal{T} \mid \underline{x} \mid \underline{\mu x.A} \mid A \times A \mid A \to A$$

- $\mu x.T$ binds x in T, free and bound variables as expected
- μ-types are closed and <u>contractive</u> terms

when are two types equal ?

$$\mu y.y \stackrel{?}{=} \mu x.z$$

$$\mu y.y \stackrel{?}{=} \mu x.x$$

$$\mu x.(int \times x) \stackrel{?}{=} int \times \mu x.(int \times x)$$

$$\mu x.x \to x \stackrel{?}{=} (\mu x.x \to x) \to (\mu x.x \to x)$$

Recursive types

$$A \;::=\; \mathcal{T} \mid \underline{x} \mid \underline{\mu x.A} \mid A \times A \mid A \to A$$

- $\mu x.T$ binds x in T, free and bound variables as expected
- μ-types are closed and <u>contractive</u> terms

when are two types equal ?

$$\mu y.y \stackrel{?}{=} \mu x.z$$ Not a type!

$$\mu y.y \stackrel{?}{=} \mu x.x$$

$$\mu x.(int \times x) \stackrel{?}{=} int \times \mu x.(int \times x)$$

$$\mu x.x \to x \stackrel{?}{=} (\mu x.x \to x) \to (\mu x.x \to x)$$

Type equivalence semantic approach

Σ: set of symbols with an arity — ranked alphabet

A *tree* over a ranked alphabet Σ is a partial function $t : \mathbb{N}_+^\star \to \Sigma$ such that

- $dom(t)$ non-empty
- $dom(t)$ prefix-closed
- for all $\pi \in dom(t)$
  - $i, j \in N_+^\star, 1 \leq i \leq j$ and $\pi j \in dom(t)$ imply $\pi i \in dom(t)$
  - $t(\pi) = A$ of arity $k \geq 0$ implies for $i \in \mathbb{N}_+, \pi i \in dom(t)$ iff $1 \leq i \leq k$

Extensional equivalence (naïve)

- f, g functions
- $f \overset{ext}{=} g$ if $dom(f) = dom(g)$ and $\forall x \in dom(f).\, f(x) = g(x)$

Type equivalence semantic approach

$$\Sigma = \mathcal{T} \cup \{\times, \rightarrow\}$$

$$\begin{array}{lcll}
treeof(c)(\varepsilon) & = & c & \text{where } c \in \mathcal{T} \\
treeof(A_1 \rightarrow A_2)(\varepsilon) & = & \rightarrow & \\
treeof(A_1 \rightarrow A_2)(i\pi) & = & treeof(A_i)(\pi) & \\
\vdots & & & \\
treeof(\mu x.A)(\pi) & = & treeof(A\{x/\mu x.A\})(\pi) &
\end{array}$$

Lemma

For every μ-type A the $treeof(A)$ is defined. **Why ?** □

Let $A \stackrel{ext}{=} B$ whenever $treeof(A) \stackrel{ext}{=} treeof(B)$

Type equivalence semantic approach

$$\Sigma = \mathcal{T} \cup \{\times, \rightarrow\}$$

$$\begin{array}{lcll}
\mathit{treeof}(c)(\varepsilon) & = & c & \text{where } c \in \mathcal{T} \\
\mathit{treeof}(A_1 \rightarrow A_2)(\varepsilon) & = & \rightarrow & \\
\mathit{treeof}(A_1 \rightarrow A_2)(i\pi) & = & \mathit{treeof}(A_i)(\pi) & \\
\vdots & & & \\
\mathit{treeof}(\mu x.A)(\pi) & = & \mathit{treeof}(A\{x/\mu x.A\})(\pi) &
\end{array}$$

Lemma

For every μ-type A the $\mathit{treeof}(A)$ is defined. **Why ?** $\square$

Let $A \stackrel{ext}{=} B$ whenever $\mathit{treeof}(A) \stackrel{ext}{=} \mathit{treeof}(B)$

$$\boxed{\text{How to decide } \stackrel{ext}{=} \text{ ?}}$$

Pour le TP

- Which option of ocaml allows to type $\mathcal{Y}$?
- Implement
  - *treeof* easy
  - *fact* as least fixed point think of $\mathcal{Y}$, `fix`,...
  - algorithm to decide equality over recursive types hard