Typage

Proof-theoretic approach to (co)induction

2018-2019

Giovanni Bernardi, `gioXYZirif.fr`

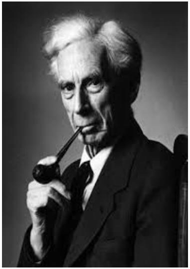
`http://www.irif.fr/~gio/index.xhtml`

Université Paris Diderot

Plan

1. Historical remark
2. Recap a few points
3. Questions
4. Proof theoretic approach — and its set-theoretic explanation
5. Examples examples examples

1908, Russell

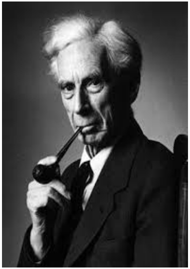


These fallacies [. . .] are to be avoided by what may be called the "vicious-circle principle;" *i.e.*, [. . .] whatever contains an apparent variable must be of a different type from the possible values of that variable [. . .] This is the guiding principle in what follows.

1968, Morris

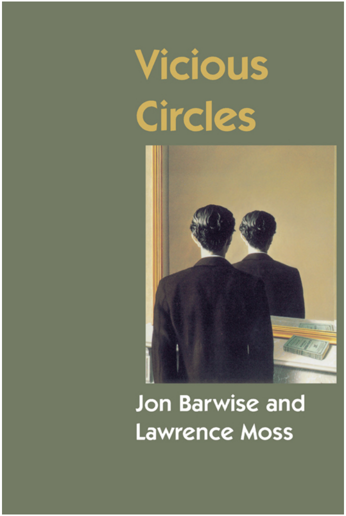
This construction is shown to be lacking [. . .] the type system makes the λ-calculus an uninteresting programming language; i.e. one without non-terminating computations.

1908, Russell



1968, Morris

1996



Thus far ...

Motivated by circularities, we discussed

Theory

1. Functions over partial orders F, $\langle P, \leq \rangle$
2. Fixed points $x = F(x)$

-	least	induction	Kleene fp theorem	μF
-	greatest	coinduction	Knaster-Tarski theorem	νF

Applications

- Subtyping / equality for recursive types
- Equi-recursive type system how to type $\mathcal{Y}$

Recap: relations

- Assuming sets, $\subseteq$, $\in$
- $X \times Y = \{\, (x, y) \mid \text{ all } x \in X \text{ and } y \in \mathcal{Y} \,\}$ Cartesian product
- $parts(X) = \{\, Z \mid Z \subseteq X \,\}$ powerset
- A *relation* R between sets X and Y is a subset of $X \times Y$
  - $R \in parts(X \times Y)$
  - Notation: $x\, R\, y$ means $(x, y) \in R$
- A relation $R \subseteq X \times X$ is
  - *reflexive* if $x\, R\, x$ $\forall x \in X$
  - *symmetric* if $x\, R\, y$ implies $y\, R\, x$ $\forall x, y \in X$
  - *antisymmetric* if $x\, R\, y$ and $y\, R\, x$ imply $x = y$ $\forall x, y \in X$
  - *transitive* if $x\, R\, y$ and $y\, R\, z$ imply $x\, R\, z$ $\forall x, y, z \in X$
  - *total* if $x\, R\, y$ or $y\, R\, x$ for every $x, y \in X$
  - a *preorder* if it is reflexive and transitive
  - a **partial order** if it is reflexive, antisymmetric, and transitive
  - an *equivalence* if is reflexive, symmetric, and transitive

Recap: orders

- Notation: $\langle P, \leq \rangle$ where P set and $\leq \subseteq P \times P$ partial order
- $\langle P, \leq \rangle$ partially ordered set: **poset**
- If $\langle P, \leq \rangle$ poset and $S \subseteq P$
  - $S^u = \{ x \in P \mid \forall s \in S.\, s \leq x \}$ — S upper
  - $x \in S^u$ is an upper bound of S — $\forall x$
  - $x \in S^u$ is the least upper bound of S if $\forall y \in S^u.\, x \leq y$ — $\forall x$
  - $\bigsqcup S$ denotes the **least upper bound** of S

  - $S^\ell = \{ x \in P \mid \forall s \in S.\, x \leq s \}$ — S lower
  - $x \in S^\ell$ is an lower bound of S — $\forall x$
  - $x \in S^\ell$ is the greatest lower bound of S if $\forall y \in S^\ell.\, y \leq x$ — $\forall x$
  - $\bigsqcap S$ denotes the **greatest lower bound** of S

λ-calculus

typing rules from [Cardone and Coppo, 1991]

$$M, N ::= x \mid c \mid MN \mid \lambda x.M$$

An equi-recursive system

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{}{\Gamma, g : \mathit{typeof}(g) \vdash g : \mathit{typeof}(g)}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

$$\frac{\Gamma \vdash M : B}{\Gamma \vdash M : A} \; A \approx B$$

Powerful type system, for instance we can type $\mathcal{Y}$

Type equivalence syntactic approach

$$
\begin{array}{lcl}
F & : & \mathit{parts}(\mathrm{Types}_\mu^2) \to \mathit{parts}(\mathrm{Types}_\mu^2) \\
F(\mathcal{R}) & \triangleq & \{\, (c, c) \mid c \in \mathcal{T} \,\} \\
& & \cup \{\, (A_1 \times A_2, B_1 \times B_2) \mid \forall i \in \{1,2\}.\, A_i \;\mathcal{R}\; B_i \,\} \\
& & \cup \{\, (A_1 \to A_2, B_1 \to B_2) \mid B_1 \;\mathcal{R}\; A_1, A_2 \;\mathcal{R}\; B_2 \,\} \\
& & \cup \{\, (A, \mu x.B) \mid A \;\mathcal{R}\; B\{x/\mu x.B\} \,\} \\
& & \cup \{\, (\mu x.A, B) \mid A\{x/\mu x.A\} \;\mathcal{R}\; B \,\}
\end{array}
$$

We have

- $\langle \mathit{parts}(\mathrm{Types}_\mu^2), \subseteq \rangle$ complete lattice, F monotone
- $\nu F = \bigcup \{\, \mathcal{R} \in \mathit{parts}(\mathrm{Types}_\mu^2) \mid \mathcal{R} \subseteq F(\mathcal{R}) \,\}$ by Knaster-Tarski
- Let $\leq_{sbt}^{c} \triangleq \nu F$ and $\approx \triangleq \leq_{sbt}^{c} \cap (\leq_{sbt}^{c})^{-1}$

Questions questions questions ...

1. What is a complete lattice?

Questions questions questions . . .

1. What is a complete lattice?
2. What is a complete partial order (CPO) ?

Questions questions questions . . .

1. What is a complete lattice?
2. What is a complete partial order (CPO) ?
3. What does the Knaster-Tarski theorem state ?

Questions questions questions . . .

1. What is a complete lattice?
2. What is a complete partial order (CPO) ?
3. What does the Knaster-Tarski theorem state ?
4. What does Kleene fixed point theorem state ?

Let's change perspective

inference rule

$$\frac{\text{premise}_1 \quad \ldots \quad \text{premise}_n}{\text{conclusion}} \ \text{side condition}$$

example

$$\frac{\Gamma \vdash M : B}{\Gamma \vdash M : A} \ A \approx B$$

Back to non-recursive types

Minimal language of types $\quad A, B \quad ::= \quad int \mid real \mid A \rightarrow A$

Subtyping relation ground types

$$int \leq_g int \qquad real \leq_g real \qquad int \leq_g real$$

How to define subtyping $\leq_{sbt}$ on types $A, B, \ldots$?

Back to non-recursive types

Minimal language of types $\quad A, B \;::=\; int \mid real \mid A \to A$
Subtyping relation ground types

$$int \leq_g int \qquad real \leq_g real \qquad int \leq_g real$$

How to define subtyping $\leq_{sbt}$ on types $A, B, \ldots$?

inference rules

$$\frac{}{c_1 \leq_{sbt} c_2}\; c_1 \leq_g c_2 \qquad\qquad \frac{B_1 \leq_{sbt} A_1 \quad A_2 \leq_{sbt} B_2}{A_1 \to A_2 \leq_{sbt} B_1 \to B_2}$$

Inductive definition

Relation $\leq_{sbt}$ contains all pairs (A, B) s.t. $\qquad$ set theoretic ideas

- we can **derive** $A \leq_{sbt} B$,
- via a **finite derivation tree**

Back to non-recursive types

A derivation tree of depth 2 (i.e. finite)

$$\dfrac{\dfrac{}{int \leq_{sbt} real} \qquad \dfrac{}{int \leq_{sbt} real}}{real \to int \leq_{sbt} int \to real}$$

$$\dfrac{}{c_1 \leq_{sbt} c_2}\ c_1 \leq_g c_2 \qquad\qquad \dfrac{B_1 \leq_{sbt} A_1 \quad A_2 \leq_{sbt} B_2}{A_1 \to A_2 \leq_{sbt} B_1 \to B_2}$$

Inductive definition

Relation $\leq_{sbt}$ contains all pairs (A, B) s.t. set theoretic ideas

- we can **derive** $A \leq_{sbt} B$,
- via a **finite derivation tree**

Back to non-recursive types

A derivation tree of depth 2 (i.e. finite)

$$\dfrac{\dfrac{}{\mathit{int} \leq_{sbt} \mathit{real}} \qquad \dfrac{}{\mathit{int} \leq_{sbt} \mathit{real}}}{\mathit{real} \to \mathit{int} \leq_{sbt} \mathit{int} \to \mathit{real}}$$

$$\dfrac{}{c_1 \leq_{sbt} c_2}\ c_1 \leq_g c_2 \qquad\qquad \dfrac{B_1 \leq_{sbt} A_1 \quad A_2 \leq_{sbt} B_2}{A_1 \to A_2 \leq_{sbt} B_1 \to B_2}$$

Inductive definition

How to express this using sets/functions ?

From rules to functions

inference rules

$$\frac{}{c_1 \leq_{sbt} c_2}\; c_1 \leq_g c_2 \qquad\qquad \frac{B_1 \leq_{sbt} A_1 \quad A_2 \leq_{sbt} B_2}{A_1 \to A_2 \leq_{sbt} B_1 \to B_2}$$

What do the rules *mean*?

From rules to functions

inference rules

$$\frac{}{(c_1, c_2)}\ c_1 \leq_g c_2 \qquad\qquad \frac{(B_1, A_1) \quad (A_2, B_2)}{(A_1 \to A_2, B_1 \to B_2)}$$

What do the rules *mean*?

To define a binary relation $\leq_{sbt}$

From rules to functions

inference rules

$$\frac{}{(c_1, c_2)}\ c_1 \leq_g c_2 \qquad\qquad \frac{(B_1, A_1) \quad (A_2, B_2)}{(A_1 \to A_2, B_1 \to B_2)}$$

What do the rules *mean*?

To define a binary relation $\leq_{sbt}$, the rules define

$$F \quad : \quad \mathit{parts}(\mathrm{Types}^2) \to \mathit{parts}(\mathrm{Types}^2)$$

$$\begin{aligned} F(\mathcal{R}) \quad &\triangleq \quad \{\, (c_1, c_2) \mid c_1 \leq_g c_2 \,\} \\ &\quad \cup \{\, (A_1 \to A_2, B_1 \to B_2) \mid B_1 \mathrel{\mathcal{R}} A_1, A_2 \mathrel{\mathcal{R}} B_2 \,\} \end{aligned}$$

From rules to functions

inference rules

$$\boxed{\frac{}{(c_1, c_2)}\ c_1 \le_g c_2 \qquad\qquad \frac{(B_1, A_1) \quad (A_2, B_2)}{(A_1 \to A_2, B_1 \to B_2)}}$$

What do the rules *mean*?

To define a binary relation $\le_{sbt}$, the rules define

$$\begin{array}{lcl} F & : & \mathit{parts}(\mathrm{Types}^2) \to \mathit{parts}(\mathrm{Types}^2) \\ F(\mathcal{R}) & \triangleq & \{(\mathit{int}, \mathit{int}), (\mathit{real}, \mathit{real}), (\mathit{int}, \mathit{real})\} \\ & & \cup \{\, (A_1 \to A_2, B_1 \to B_2) \mid B_1 \,\mathcal{R}\, A_1, A_2 \,\mathcal{R}\, B_2 \,\} \end{array}$$

From derivation trees to function application

$$
\begin{aligned}
F(\mathcal{R}) \;\; &\triangleq \;\; \{(\mathit{int}, \mathit{int}), (\mathit{real}, \mathit{real}), (\mathit{int}, \mathit{real})\} \\
&\quad \cup \{\, (A_1 \to A_2, B_1 \to B_2) \;|\; B_1 \; \mathcal{R} \; A_1, A_2 \; \mathcal{R} \; B_2 \,\}
\end{aligned}
$$

Let's use F,

$$
\begin{aligned}
F^0(\emptyset) \;\; &= \;\; \emptyset \qquad\qquad\qquad \textit{by convention} \\
F^1(\emptyset) \;\; &= \\
F^2(\emptyset) \;\; &=
\end{aligned}
$$

From derivation trees to function application

$$
\begin{aligned}
F(\mathcal{R}) \;\triangleq\;& \{(\mathit{int}, \mathit{int}), (\mathit{real}, \mathit{real}), (\mathit{int}, \mathit{real})\} \\
& \cup \{\, (A_1 \to A_2, B_1 \to B_2) \;\mid\; B_1 \mathrel{\mathcal{R}} A_1, A_2 \mathrel{\mathcal{R}} B_2 \,\}
\end{aligned}
$$

Let's use F,

$$
\begin{aligned}
F^0(\emptyset) &= \emptyset && \textit{by convention} \\
F^1(\emptyset) &= \{(\mathit{int}, \mathit{int}), (\mathit{real}, \mathit{real}), (\mathit{int}, \mathit{real})\} = \;\leq_g \\
F^2(\emptyset) &=
\end{aligned}
$$

From derivation trees to function application

$$
\begin{aligned}
F(\mathcal{R}) \;\triangleq\;& \{(int, int), (real, real), (int, real)\} \\
& \cup \{\, (A_1 \to A_2, B_1 \to B_2) \mid B_1 \mathrel{\mathcal{R}} A_1, A_2 \mathrel{\mathcal{R}} B_2 \,\}
\end{aligned}
$$

Let's use F,

$$
\begin{aligned}
F^0(\emptyset) &= \emptyset && \textit{by convention} \\
F^1(\emptyset) &= \{(int, int), (real, real), (int, real)\} = \;\leq_g \\
F^2(\emptyset) &= \{(real \to int, int \to real), (int \to int, int \to int), \ldots\} \\
& \quad \cup \leq_g \\
\vdots & \qquad \vdots
\end{aligned}
$$

From derivation trees to function application

The same derivation tree of depth 2

$$\dfrac{\dfrac{}{(int, real)} \qquad \dfrac{}{(int, real)}}{(real \to int, int \to real)}$$

$$\begin{array}{lcll}
F^0(\emptyset) & = & \emptyset & \textit{by convention} \\
F^1(\emptyset) & = & \{(int, int), (real, real), (int, real)\} = \ \leq_g & \\
F^2(\emptyset) & = & \{(real \to int, int \to real), (int \to int, int \to int), \ldots\} & \\
 & & \cup \leq_g & \\
\vdots & & \vdots &
\end{array}$$

From derivation trees to function application

Definition

Relation $\leq_{sbt}$ contains all pairs (A, B) s.t.

- we can **derive** $A \leq_{sbt} B$,
- via a **finite derivation tree**

Lemma

A derivation tree $\dfrac{\vdots}{(A, B)}$ *has depth* n *iff* $(A, B) \in F^n(\emptyset)$. $\square$

but then ...

From derivation trees to function application

Definition

Relation $\leq_{sbt}$ contains all pairs (A, B) s.t.

- we can **derive** $A \leq_{sbt} B$,
- via a **finite derivation tree**

Lemma

A derivation tree $\dfrac{\vdots}{(A, B)}$ *has depth* n *iff* $(A, B) \in F^{n}(\emptyset)$. $\square$

but then ...

Corollary

$\leq_{sbt} = \bigcup_{n=0} F^{n}(\emptyset)$, thus by Kleene fixed point theorem

$$\boxed{\leq_{sbt} = \mu F}$$

Recursive types

$$A \quad ::= \quad int \mid real \mid x \mid \mu x.A \mid A \to A$$

$$\begin{aligned}
F \quad &: \quad parts(\mathrm{Types}_\mu^2) \to parts(\mathrm{Types}_\mu^2) \\
F(\mathcal{R}) \quad &\triangleq \quad \{(int, int), (real, real), (int, real)\} \\
&\quad \cup \{\, (A_1 \to A_2, B_1 \to B_2) \mid B_1 \mathrel{\mathcal{R}} A_1, A_2 \mathrel{\mathcal{R}} B_2 \,\} \\
&\quad \cup \{\, (A, \mu x.B) \mid A \mathrel{\mathcal{R}} B\{x/\mu x.B\} \,\} \\
&\quad \cup \{\, (\mu x.A, B) \mid A\{x/\mu x.A\} \mathrel{\mathcal{R}} B \,\}
\end{aligned}$$

- $\langle parts(\mathrm{Types}_\mu^2), \subseteq \rangle$ complete lattice, F monotone
- νF exists by Knaster-Tarski
- Let $\leq_{sbt}^{c} \triangleq \nu F$, $\quad \approx \triangleq \leq_{sbt}^{c} \cap (\leq_{sbt}^{c})^{-1}$

Recursive types

$$A \ ::= \ int \mid real \mid x \mid \mu x.A \mid A \to A$$

inference rules

$$\frac{}{c_1 \leq'_{sbt} c_2}\ c_1 \leq_g c_2 \qquad \frac{B_1 \leq'_{sbt} A_1 \quad A_2 \leq'_{sbt} B_2}{A_1 \to A_2 \leq'_{sbt} B_1 \to B_2}$$

$$\frac{A \leq'_{sbt} B\{x/\mu x.B\}}{A \leq'_{sbt} \mu x.B} \qquad \frac{A\{x/\mu x.A\} \leq'_{sbt} B}{\mu x.A \leq'_{sbt} B}$$

Coinductive definition

Relation $\leq'_{sbt}$ contains all pairs (A, B) s.t.

- we can derive $A \leq'_{sbt} B$
- via a finite **or a circular** derivation tree

A circular derivation tree

Example

Let $A = \mu x.x \to int$, let's show that $A \leq'_{sbt} A \to int$.

$$
\dfrac{\dfrac{\dfrac{}{A \leq'_{sbt} A \to int}}{A \leq'_{sbt} A} \qquad \dfrac{}{int \leq'_{sbt} int}}{\dfrac{A \to int \leq'_{sbt} A \to int}{A \leq'_{sbt} A \to int}}
$$

A circular derivation tree

Example

Let $A = \mu x.x \to int$, let's show that $A \leq'_{sbt} A \to int$.

$$\dfrac{\dfrac{\dfrac{}{A \leq'_{sbt} A \to int}}{A \leq'_{sbt} A} \qquad \dfrac{}{int \leq'_{sbt} int}}{\dfrac{A \to int \leq'_{sbt} A \to int}{A \leq'_{sbt} A \to int}}$$

$\boxed{\text{What's the relation with } \nu F \text{ ??}}$

A circular derivation tree

Example

Let $A = \mu x.x \to int$, let's show that $A \leq'_{sbt} A \to int$.

$$\cfrac{\cfrac{\cfrac{}{A \leq'_{sbt} A \to int}}{A \leq'_{sbt} A} \qquad \cfrac{}{int \leq'_{sbt} int}}{\cfrac{A \to int \leq'_{sbt} A \to int}{A \leq'_{sbt} A \to int}}$$

$\boxed{\text{What's the relation with } \nu F \text{ ??}}$

- $\mathcal{R} \triangleq \{(A, A \to int), (A \to int, A \to int), (A, A), (int, int)\}$
- $\mathcal{R} \subseteq F(\mathcal{R})$ post-fixed point
- $\mathcal{R} \subseteq \nu F = \leq^c_{sbt}$
- In fact we have $\leq^c_{sbt} = \leq'_{sbt}$

Summary

Induction

- least fixed points — Kleene fp theorem
- finite derivation trees

Coinduction

- greatest fixed points — Knaster-Tarski fp theorem
- finite and circular derivation trees

Example

Subtyping relation

Other more abstract approaches exist — category theory


That's all Folks!