Typage

Coinduction

2018-2019
Giovanni Bernardi, gioXYZirif.fr
http://www.irif.fr/~gio/index.xhtml
Université Paris Diderot
Plan

1. Questions
2. Mini historical remarks
3. More fixed points
4. Deciding type equivalence
5. A type system with recursive types
1. In which year was the first paper on types published?
1. In which year was the first paper on types published?
2. What does Kleene fixed point theorem state?
Questions questions questions . . .

1. In which year was the first paper on types published?
2. What does Kleene fixed point theorem state?
3. What is a tree?
1. In which year was the first paper on types published?
2. What does Kleene fixed point theorem state?
3. What is a tree?

\[ \Sigma = \{a, b, f, k, h\} \]

\[
s(\varepsilon) = f \\
 s(1) = s(11) = k \quad s(111) = a \\
 s(2) = h \quad s(21) = b
\]

```
      f
     / \   \\
    k    h
   /     /
  k     b
 /     /
a
```
1. In which year was the first paper on types published?
2. What does Kleene fixed point theorem state?
3. What is a tree?

Σ = \{a, b, f, k, h\}

\begin{align*}
    s(ε) &= f \\
    s(1) &= s(11) = k & s(111) &= a \\
    s(2) &= h & s(21) &= b
\end{align*}

\text{what about the arities?}
1908, Russell

A *type* is defined as the range of significance of a propositional function, *i.e.* as the collection of arguments for which that said function has values.

1968, Morris

[...] types and type declarations are often described as communications to a compiler to aid it in allocating storage, etc.
What was the problem again?

\[ A ::= \; \mathcal{T} \; | \; x \; | \; \mu x.A \; | \; A \times A \; | \; A \rightarrow A \]

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

when are two types equal?

\[
\begin{align*}
\mu y.y & \overset{?}{=} \mu x.z \\
\mu y.y & \overset{?}{=} \mu x.x \\
\mu x.(\text{int } \times x) & \overset{?}{=} \text{int } \times \mu x.(\text{int } \times x) \\
\mu x.x \rightarrow x & \overset{?}{=} (\mu x.x \rightarrow x) \rightarrow (\mu x.x \rightarrow x)
\end{align*}
\]
What was the problem again?

\[ A ::= \top | x | \mu x.A | A \times A | A \to A \]

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

\( A \) **contractive** if for any subexpression of \( A \) of the form

\[ \mu x.\mu x_1.\mu x_2.\ldots.\mu x_n.B \]

the term \( B \) is not \( x \).

- not contractive: \( \mu x.x \)
- contractive: \( \mu x.y \) **but not closed**
- not contractive: \( \text{int} \to \mu x.x \)
- contractive: \( \mu x.x \to x \)
Type equivalence semantic approach

\[
\Sigma = \mathcal{T} \cup \{ \times, \to \}
\]

\[
\text{treeof}(c)(\varepsilon) = c \quad \text{where} \quad c \in \mathcal{T}
\]
\[
\text{treeof}(A_1 \to A_2)(\varepsilon) = \to
\]
\[
\text{treeof}(A_1 \to A_2)(i\pi) = \text{treeof}(A_i)(\pi)
\]
\[
\vdots
\]
\[
\text{treeof}(\mu x.A)(\pi) = \text{treeof}(A\{x/\mu x.A\})(\pi)
\]

**Lemma**

*For every \(\mu\)-type \(A\) the treeof\((A)\) is defined.*  
**Why?**

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

How to decide \(\stackrel{\text{ext}}{=} \) ?
More on fixed points

Theorem (Knaster 1928 - Tarski 1955)

If \( \langle L, \leq \rangle \) complete lattice, \( f : L \rightarrow L \) monotone function then

\[ \mu f = \bigcap \{ x \mid f(x) \leq x \} \]
\[ \nu f = \bigcup \{ x \mid x \leq f(x) \} \]
More on fixed points

A poset $\langle L, \leq \rangle$ is a complete lattice if

- $L \neq \emptyset$, and
- for every $S \in \text{parts}(L)$. $\bigsqcup S$ and $\bigcap S$ exist

Lemma

*Every complete lattice is a CPO.*

Theorem (Knaster 1928 - Tarski 1955)

If $\langle L, \leq \rangle$ complete lattice, $f : L \to L$ monotone function then

- $\mu f = \bigcap \{ x \mid f(x) \leq x \}$
- $\nu f = \bigsqcup \{ x \mid x \leq f(x) \}$

\[ \square \]
More on fixed points

A poset \( \langle L, \leq \rangle \) is a complete lattice if
\begin{itemize}
  \item \( L \neq \emptyset \), and
  \item for every \( S \in \text{parts}(L) \). \( \sqcup S \) and \( \sqcap S \) exist
\end{itemize}

Lemma
Every complete lattice is a CPO.

Theorem (Knaster 1928 - Tarski 1955)
If \( \langle L, \leq \rangle \) complete lattice, \( f : L \to L \) monotone function then
\begin{itemize}
  \item \( \mu f = \sqcap \{ x \mid f(x) \leq x \} \)
  \item \( \nu f = \sqcup \{ x \mid x \leq f(x) \} \quad \text{coinduction} \)
\end{itemize}
Type equivalence syntactic approach

\[ F : \text{parts}(\text{Types}_\mu^2) \rightarrow \text{parts}(\text{Types}_\mu^2) \]

\[ F(\mathcal{R}) \overset{\Delta}{=} \{ (c, c) \mid c \in \mathcal{T} \} \]
\[ \cup \{ (A_1 \times A_2, B_1 \times B_2) \mid \forall i \in \{1, 2\}. A_i \mathcal{R} B_i \} \]
\[ \cup \{ (A_1 \rightarrow A_2, B_1 \rightarrow B_2) \mid B_1 \mathcal{R} A_1, A_2 \mathcal{R} B_2 \} \]
\[ \cup \{ (A, \mu x.B) \mid A \mathcal{R} B\{x/\mu x.B\} \} \]
\[ \cup \{ (\mu x.A, B) \mid A\{x/\mu x.A\} \mathcal{R} B \} \]

\[ \langle \text{parts}(\text{Types}_\mu^2), \subseteq \rangle \text{ complete lattice, } F \text{ monotone} \]

\[ \nu F \text{ exists} \quad \text{by Knaster-Tarski} \]

\[ \leq : \overset{\Delta}{=} \nu F \]
\[ \approx \overset{\Delta}{=} \leq : \cap \leq :^{-1} \]
Type equivalence

Syntactic definition justified by semantic one

\[ \approx = \text{ext} \]

How to show \( A \approx B \)? Show \( A \prec B \) and \( B \prec A \) — no brainer

Coinductive proof method

How to show \( A \prec B \) ?

1. By definition \( \prec = \nu F \)
2. By Knaster-Tarski \( \prec = \bigcup \{ \mathcal{R} \mid \mathcal{R} \subseteq F(\mathcal{R}) \} \)
3. It suffices to define relation \( \mathcal{R} \) such that

\[ A \mathcal{R} B, \quad \mathcal{R} \subseteq F(\mathcal{R}) \]
Example

Let \( A = \mu x . x \rightarrow x \), why \( A \approx A \rightarrow A \)?

Let

\[
\mathcal{R} = \{(A, A \rightarrow A)\}
\]

1. By definition \( A \mathcal{R} A \rightarrow A \)

2. Routine work shows that \( \mathcal{R} \subseteq F(\mathcal{R}) \) and \( \mathcal{R}^{-1} \subseteq F(\mathcal{R}^{-1}) \),

\[
A <: A \rightarrow A, \quad A \rightarrow A <: A
\]
Example

Let \( A = \mu x . x \to x \), why \( A \approx A \to A \) ?

Let

\[
\mathcal{R} = \{(A, A \to A), (A \to A, A \to A), \}
\]

1. By definition \( A \mathcal{R} A \to A \)

2. Routine work shows that \( \mathcal{R} \subseteq F(\mathcal{R}) \) and \( \mathcal{R}^{-1} \subseteq F(\mathcal{R}^{-1}) \),

\[
A <: A \to A, \quad A \to A <: A
\]
Example

Let \( A = \mu x.x \to x \), why \( A \approx A \to A \) ?

Let

\[
\mathcal{R} = \{(A, A \to A), (A \to A, A \to A), (A, A)\}
\]

1. By definition \( A \mathcal{R} A \to A \)
2. Routine work shows that \( \mathcal{R} \subseteq F(\mathcal{R}) \) and \( \mathcal{R}^{-1} \subseteq F(\mathcal{R}^{-1}) \),

\[
A <: A \to A, \quad A \to A <: A
\]
Example

Let \( A = \mu x. x \to x \), why \( A \approx A \to A \)?

Let

\[
\mathcal{R} = \{(A, A \to A), (A \to A, A \to A), (A \to A, A), (A, A)\}
\]

1. By definition \( A \mathcal{R} A \to A \)
2. Routine work shows that \( \mathcal{R} \subseteq F(\mathcal{R}) \) and \( \mathcal{R}^{-1} \subseteq F(\mathcal{R}^{-1}) \),

\[
A <: A \to A, \quad A \to A <: A
\]
Example

Let $A = \mu x. x \to x$, why $A \approx A \to A$?

Let

$$\mathcal{R} = \{(A, A \to A), (A \to A, A \to A), (A \to A, A), (A, A)\}$$

1. By definition $A \mathcal{R} A \to A$
2. Routine work shows that $\mathcal{R} \subseteq F(\mathcal{R})$ and $\mathcal{R}^{-1} \subseteq F(\mathcal{R}^{-1})$, $A <: A \to A$, $A \to A <: A$

Write a decision procedure for $\approx$
\[
M, N ::= x \mid c \mid MN \mid \lambda x. M
\]

An equi-recursive system

\[
\Gamma, x : A \vdash x : A \\
\Gamma, c : A \vdash c : A_c(c)
\]

\[
\begin{array}{c}
\Gamma, x : A \vdash M : B \\
\hline
\Gamma \vdash \lambda x. M : A \rightarrow B
\end{array}
\]

\[
\begin{array}{c}
\Gamma \vdash M : A \rightarrow B \\
\hline
\Gamma \vdash N : A
\end{array}
\]

\[
\begin{array}{c}
\Gamma \vdash M : B \\
\Gamma \vdash M : A
\end{array}
\]

\[
A \approx B
\]
Example

Let $A = \mu x.((x \to x) \to x)$

\[
\begin{align*}
\frac{x : A \to A \vdash x : A \to A}{x : A \to A \vdash x : A \to A} & \quad \frac{x : A \to A \vdash x : A \to A}{x : A \to A \vdash x : A} \\
\frac{x : A \to A \vdash xx : (A \to A) \to A}{(\approx)} & \quad \frac{x : A \to A \vdash xx : A}{(\approx)} \\
\vdash \lambda x.xx : (A \to A) \to A
\end{align*}
\]

\[
\begin{align*}
\frac{x : A \vdash x : A}{(\approx)} & \quad \frac{x : A \vdash x : A}{(\approx)} \\
\frac{x : A \vdash xx : ((A \to A) \to A)}{(\approx)} & \quad \frac{x : A \vdash x : A}{(\approx)} \\
\frac{x : A \vdash xx : (A \to A) \to A}{(\approx)} & \vdash \lambda x.xx : A \\
\vdash (\lambda x.xx)(\lambda x.xx) : (A \to A) \to A
\end{align*}
\]

\[\]
λ-calculus

typing rules from [Cardone and Coppo, 1991]

An equi-recursive system

\[ \Gamma, x : A \vdash x : A \]
\[ \Gamma, c : A \vdash c : A_c(c) \]

\[ \Gamma, x : A \vdash M : B \]
\[ \Gamma \vdash \lambda x. M : A \rightarrow B \]

\[ \Gamma \vdash M : A \rightarrow B \]
\[ \Gamma \vdash N : A \]
\[ \Gamma \vdash MN : B \]

\[ \Gamma \vdash M : B \]
\[ \Gamma \vdash M : A \]
\[ A \approx B \]

★ Strong Normalisation is false!

\[ \vdash (\lambda x.(xx))(\lambda x.(xx)) \]
Implement

- `treeof`
- decision procedure for \( \approx \)
- Work on the project