Info

- https://www.irif.fr/~gio/index.xhtml
- gio (vous-savez-quoi) irif.fr
- subject: “[ typage ] ... ”
- Questions? office 4026
- Stop me at 11:30 !!!! il y a le TP!!
Check on-line

?? 21 Mars ??
Aim

introduction to coinduction

with an application to 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? \[ A \triangleq \{ x \mid x \not\in x \} \]

Who brought types into “PL”? 
Who conceived types?

Mathematical Logic as Based on the Theory of Types
Bertrand Russell, 1908

Why? $A \triangleq \{ x \mid x \not\in x \}$

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? \[ A \triangleq \{ x \mid x \not\in x \} \]

annus mirabilis CS

1936

Types are older than CS
Non trivial phenomenon

\[ \text{fact} \triangleq \lambda x. \text{if } x = 0 \text{ then } 1 \text{ else } x \ast (\text{fact}(x - 1)) \]

\[ \text{List 'a} \triangleq [\ ] \mid 'a : \text{List 'a} \]

\[ M, N ::= x \mid \lambda x. M \mid MN \]
Non trivial phenomenon

\[ \text{fact} \triangleq \lambda x. \text{if } x = 0 \text{ then } 1 \text{ else } x \ast (\text{fact}(x - 1)) \]

\[ \text{List }'a \triangleq [] \mid 'a : \text{List }'a \]

\[ M, N ::= x \mid \lambda x. M \mid MN \]
Non trivial phenomenon

\[ \text{fact} \triangleq \lambda x. \text{if } x = 0 \text{ then } 1 \text{ else } x \ast (\text{fact}(x - 1)) \]

\[ \text{List}'a \triangleq [] \mid 'a:\text{List}'a \]

\[ M, N ::= x \mid \lambda x. M \mid MN \]

How to treat with circularity?

structure

\[ x = F(x) \]

\[ x \text{ fixed point of } F \]

least fixed points  induction  recursion

least fixed points  coinduction  corecursion
Consider the following circular definitions

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

**Intuition**

\[ a = b \]

- **How to prove** \( a = b \)?
  - **Hint**: where is the base case in the definitions?

- **How to define** \( = \) over lists?
Theorem (Kleene, 1936)

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

$\square$
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 $\bot$ element
  - $\bigcup D$ 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
- $f(\bigcup D) = \bigcup f(D)$

**Theorem (Kleene, 1936)**

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

**Theorem (Kleene, 1936)**

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


Induction hunder the hood: set-theoretic approac

Typial CPO: powerset

\[ S = \{a, b, c\} \]

\[
\begin{array}{ccc}
S & | & \\
\{a, b\} & | & \{a, c\} & | & \{b, c\} \\
| & \times & \times & | \\
\{a\} & | & \{b\} & | & \{c\} \\
| & \times & | \\
\emptyset & | \\
\end{array}
\]

Hasse diagram of set inclusion.

**Theorem (Kleene, 1936)**

Let \( \langle P, \leq \rangle \) CPO and \( f : P \rightarrow P \) a monotone continuous function. We have \( \mu f = \bigcup_{n \geq 0} f^n(\bot) \).
Factorial as least fixed point

\[ F \overset{\Delta}{=} \lambda y.\lambda x. \text{if } x = 0 \text{ then } 1 \text{ else } x \ast (y(x - 1)) \]

\[ F : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N} \rightarrow \mathbb{N}) \]

\[ \langle \mathbb{N} \rightarrow \mathbb{N}, \leq \rangle \text{ CPO with bottom } \emptyset \text{ and } F(y) \text{ continuous in } y, \]

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

\[ \text{NB: } \mu y. F(y) \text{ is a function!} \]
Factorial as least fixed point

\[ F \overset{\Delta}{=} \lambda y.\lambda x. \text{if } x = 0 \text{ then } 1 \text{ else } x \ast (y(x - 1)) \]

\[ F : (\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N}) \]

\[ \text{fact} \overset{\Delta}{=} \mu y. F(y) \]

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

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

\[ \text{NB: } \mu y. F(y) \text{ is a function!} \]

from “definition” to property

\[ \text{fact}(x) = \text{if } x = 0 \text{ then } 1 \text{ else } x \ast (\text{fact}(x - 1)) \]
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 \( \lambda \)-theoretic approach

**fixed-point combinator**

\[
\mathcal{Y} \triangleq \lambda f. (\lambda x. f(xx))(\lambda x. f(xx))
\]

**Theorem (Kleene, 1936, “\( \lambda \)-definability and recursiveness”)**

*For every \( \lambda \)-term \( M \) we have \( \mathcal{Y}M \beta \equiv M(\mathcal{Y}M) \).*

**Theorem (Morris, 1968, PhD thesis Corollary 7(a))**

*For every \( \lambda \)-term \( M, A \) if \( A \beta \equiv MA \) then \( \mathcal{Y}M \leq A \).*

**Factorial?**

- \( F \triangleq \lambda y. \lambda x. \text{if } x = 0 \text{ then } 1 \text{ else } x \ast (y(x - 1)) \)
- \( \mathcal{Y}F \) is a fixed point of \( F \)
- \( \mathcal{Y}F \) is the least fixed point of \( F \), \( \text{fact} \triangleq \mathcal{Y}F \)
Can $\mathcal{Y}$ be typed? intuitive argument

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

**Type derivation of sub-term of $\mathcal{Y}$**

$$
\begin{array}{c}
\Gamma \vdash f : B \rightarrow C \\
\hline
\Gamma \vdash x : A \rightarrow B \\
\Gamma \vdash \lambda x. f(xx) : A \rightarrow C
\end{array}
$$

We need a type that satisfies

$$
A = A \rightarrow B
$$
Recursive types

\[ A ::= T \mid x \mid \mu x.A \mid A \times A \mid A \to A \]

- \( \mu x. T \) 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: \( int \to \mu x.x \)
- contractive: \( \mu x.x \to x \)
Recursive types

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

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

when are two types equal?

\[
\begin{align*}
\mu y. y & \quad ? \quad \mu x. z \\
\mu y. y & \quad ? \quad \mu x. x \\
\mu x. (\text{int} \times x) & \quad ? \quad \text{int} \times \mu x. (\text{int} \times x) \\
\mu x. x \to x & \quad ? \quad (\mu x. x \to x) \to (\mu x. x \to x)
\end{align*}
\]
Recursive types

\[ A ::= T \mid x \mid \mu x. A \mid A \times A \mid A \to A \]

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

<table>
<thead>
<tr>
<th>when are two types equal?</th>
<th>μy.y ( \equiv ) μx.z ( \Rightarrow ) Not a type!</th>
</tr>
</thead>
<tbody>
<tr>
<td>μy.y ( \equiv ) μx.x</td>
<td></td>
</tr>
<tr>
<td>μx.(int ( \times ) x) ( \equiv ) int ( \times ) μx.(int ( \times ) x)</td>
<td></td>
</tr>
<tr>
<td>μx.x ( \to ) x ( \equiv ) (μx.x ( \to ) x) ( \to ) (μx.x ( \to ) x)</td>
<td></td>
</tr>
</tbody>
</table>
Type equivalence  semantic approach

Σ: set of symbols with an arity  

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

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

Extensional equivalence (naïve)

- \( f, g \) functions
  - \( f \overset{\text{ext}}{=} g \) if \( \text{dom}(f) = \text{dom}(g) \) and \( \forall x \in \text{dom}(f). f(x) = g(x) \)
Type equivalence  semantic approach

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

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

Lemma

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

Let \(A^\text{ext} = B\) whenever \(treeof(A)^\text{ext} = treeof(B)\)
Type equivalence  semantic approach

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

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

Lemma

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

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

How to decide $\overset{\text{ext}}{=} ?$
Pour le TP

- Which option of ocaml allows to type $\mathcal{Y}$?
- Implement
  - $\textit{treeof}$
  - $\textit{fact}$ as least fixed point
  - algorithm to decide equality over recursive types