Typage
Master 2: Languages et Programmation

Delia Kesner, Giovanni Bernardi

2017 - 18

IRIF

université PARIS DIDEROT PARIS 7

Schedule

?

?? 21 Mars ??

Plan

1. Questions
2. Background material
3. Mini historical remarks
4. Another fixed point theorem
5. Deciding type equivalence
6. A type system with recursive types

Questions questions questions . . .

1. In which year was the first paper on types published ?

Questions questions questions . . .

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

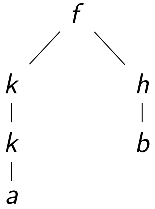
Questions questions questions . . .

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

Questions questions questions . . .

$$\Sigma = \{a, b, f, k, h\}$$

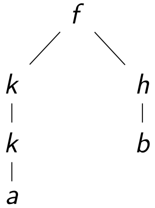
$$\begin{array}{ll} s(\varepsilon) = f & \\ s(1) = s(11) = k & s(111) = a \\ s(2) = h & s(21) = b \end{array}$$



Questions questions questions . . .

$$\Sigma = \{a, b, f, k, h\}$$

$$\begin{array}{ll} s(\varepsilon) = f & \\ s(1) = s(11) = k & s(111) = a \\ s(2) = h & s(21) = b \end{array}$$



what about the arities?

Background material: relations

- Assuming sets, $\subseteq$, $\in$
- $X \times Y = \{ (x, y) \mid \text{ all } x \in X \text{ and } y \in \mathcal{Y} \}$ Cartesian product
- $\mathcal{P}(X) = \{ Z \mid Z \subseteq X \}$ powerset
- A *relation* R between sets X and Y is a subset of $X \times Y$
  - $R \in \mathcal{P}(X \times Y)$
  - Notation: $x\,R\,y$ means $(x, y) \in R$
- A relation $R \subseteq X \times X$ is
  - *reflexive* if $x\,R\,x$ $\quad \forall x \in X$
  - *symmetric* if $x\,R\,y$ implies $y\,R\,x$ $\quad \forall x, y \in X$
  - *antisymmetric* if $x\,R\,y$ and $y\,R\,x$ imply $x = y$ $\quad \forall x, y \in X$
  - *transitive* if $x\,R\,y$ and $y\,R\,z$ imply $x\,R\,z$ $\quad \forall x, y, z \in X$
  - *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

Background material: 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

Background material: functions

- Relation $f \subseteq X \times Y$ is a partial function if

$$\forall x \in X.\, \forall y, z \in Y.\, (x, y) \in f \text{ and } (x, z) \in f \text{ imply } y = z$$

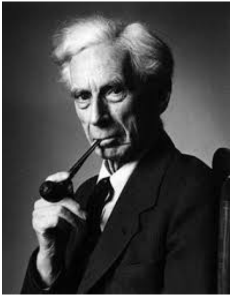
- Notation:

$$f(x) = y \text{ means } (x, y) \in f, \quad f : X \to Y \text{ means } f \subseteq X \times Y$$

- if $f : X \to Y, g : X \to Y$ then
  - $dom(f) = \{\, x \in X \mid f(x) = y \text{ for some } y \in Y \,\}$
  - $f = g$ if $dom(f) = dom(g)$ and $\forall x \in dom(f).\, f(x) = g(x)$

- If $\langle P, \leq \rangle$ poset and $f : P \to P$
  - f monotone if $x \leq y$ implies $f(x) \leq f(y)$ $\quad \forall x, y \in P$
  - x pre-fixed point of f if $f(x) \leq x$ $\quad \forall x \in P$
  - x **post-fixed point** of f if $x \leq f(x)$ $\quad \forall x \in P$
  - x **fixed point** of f if $x = f(x)$ $\quad \forall x \in P$
  - Notation: μf denotes the least fixed point of f
  - Notation: νf denotes the greatest fixed point of f

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?

Grammar for recursive types

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

- $\mathcal{T}$ set of ground types int,bool, ...
- $\mu x.A$ binds x in A, free and bound variables as expected
- $Types_\mu$ set of 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: $\mu x.x \to x$

What was the problem again?

Grammar for recursive types

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

- $\mathcal{T}$ set of ground types int,bool, ...
- $\mu x.A$ binds x in A, free and bound variables as expected
- $Types_\mu$ set of closed and <u>contractive</u> terms

when are two types equal ?

$$\boxed{\begin{array}{lcl}
\mu y.y & \stackrel{?}{=} & \mu x.z \\
\mu y.y & \stackrel{?}{=} & \mu x.x \\
\mu x.(int \times x) & \stackrel{?}{=} & int \times \mu x.(int \times x) \\
\mu x.x \rightarrow x & \stackrel{?}{=} & (\mu x.x \rightarrow x) \rightarrow (\mu x.x \rightarrow x)
\end{array}}$$

What was the problem again?

Grammar for recursive types

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

- $\mathcal{T}$ set of ground types int,bool, ...
- $\mu x.A$ binds x in A, free and bound variables as expected
- Types_μ set of closed and <u>contractive</u> terms

when are two types equal ?

$$
\boxed{
\begin{array}{lcl}
\mu y.y & \stackrel{?}{=} & \color{red}{\mu x.z \quad \text{not a type!}} \\
\mu y.y & \stackrel{?}{=} & \mu x.x \\
\mu x.(int \times x) & \stackrel{?}{=} & int \times \mu x.(int \times x) \\
\mu x.x \to x & \stackrel{?}{=} & (\mu x.x \to x) \to (\mu x.x \to x)
\end{array}
}
$$

Type equivalence — semantic approach

- $\Sigma = \mathcal{T} \cup \{\times, \rightarrow\}$ — ranked alphabet
- Trees_Σ set of trees over Σ
- Types_μ language of recursive types

Definition of $\mathit{treeof} : \mathit{Types}_\mu \rightarrow \mathit{Trees}_\Sigma$

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

Lemma
For every recursive type A the $\mathit{treeof}(A)$ is defined. $\square$

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

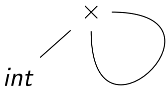
$$\boxed{\text{How to decide } \overset{ext}{=} \text{ ?}}$$

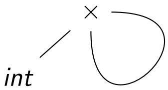
Type equivalence semantic approach

Intuitive use of *treeof*

$$\mathit{treeof}(\mu x.(\mathit{int} \times x)) \qquad \mathit{treeof}(\mathit{int} \times \mu x.(\mathit{int} \times x))$$

$$= \qquad\qquad =$$





$$\mathit{dom}(\mathit{treeof}(\mu x.(\mathit{int} \times x)) = \{\, 2^n \mid n \in \mathbb{N} \,\} \cup \{\, 2^n 1 \mid n \in \mathbb{N}_+ \,\}$$

Lemma

For every recursive type A the $\mathit{treeof}(A)$ is defined. □

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

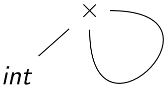
How to decide $\stackrel{ext}{=}$?

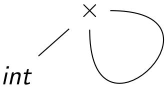
Type equivalence semantic approach

Intuitive use of *treeof*

$treeof(\mu x.(int \times x))$ $\qquad$ $treeof(int \times \mu x.(int \times x))$

$=$ $\qquad$ $=$





$dom(treeof(\mu x.(int \times x)) = \{\, 2^n \mid n \in \mathbb{N} \,\} \cup \{\, 2^n1 \mid n \in \mathbb{N}_+ \,\}$

Definition of $\stackrel{ext}{=}$ involves universal quantification!
$f = g$ if $dom(f) = dom(g)$ and $\forall x \in dom(f).f(x) = g(x)$.

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

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

Another fixed point theorem

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

- $L \neq \emptyset$, and
- for every $S \subseteq L$. $\bigsqcup S$ and $\bigsqcap S$ exist

Lemma

Every complete lattice is a CPO. $\square$

Theorem (Knaster 1928 - Tarski 1955)

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

- $\mu f = \bigsqcap \{ x \mid f(x) \leq x \}$
- $\nu f = \bigsqcup \{ x \in L \mid x \leq f(x) \}$ $\square$

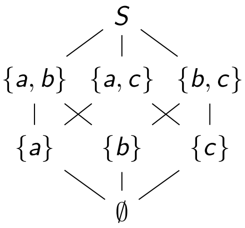
Another fixed point theorem

A poset $\langle L, \leq \rangle$ is a <u>*complete lattice*</u> if

- $L \neq \emptyset$, and
- for every $S \subseteq L$. $\bigsqcup S$ and $\bigsqcap S$ exist

A complete lattice

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



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*

- $\mu f = \bigsqcap \{ x \mid f(x) \leq x \}$
- $\nu f = \bigsqcup \{ x \in L \mid x \leq f(x) \}$ $\square$

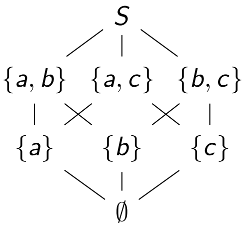
Another fixed point theorem

A poset $\langle L, \leq \rangle$ is a <u>*complete lattice*</u> if

- $L \neq \emptyset$, and
- for every $S \subseteq L$. $\bigsqcup S$ and $\bigsqcap S$ exist

A complete lattice

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



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

- $\mu f = \bigsqcap \{ x \mid f(x) \leq x \}$
- $\nu f = \bigsqcup \{ x \in L \mid x \leq f(x) \}$ *coindution* $\square$

Type equivalence syntactic approach

Let

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

We have

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

Type equivalence

Syntactic definition justified by semantic one

$$\approx\; = \;\overset{ext}{=}$$

How to show $A \approx B$? Show $A <: B$ and $B <: A$

Coinductive proof method

How to show $A <: B$?

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

$$A \,\mathcal{R}\, B, \qquad \mathcal{R} \subseteq F(\mathcal{R})$$

Example

Let $A = \mu x.x \times x$, prove that $A \approx A \times A$

Let

$$\begin{aligned} \mathcal{R} \;&=\; \{(A, A \times A), (A \times A, A \times A), \\ &\quad\; (A \times A, A), (A, A)\} \end{aligned}$$

1. By definition $A \,\mathcal{R}\, A \times A$
2. Routine work shows that $\mathcal{R} \subseteq F(\mathcal{R})$, thus
$$A <: A \times A$$
3. By definition $A \times A \,\mathcal{R}\, A$
4. Routine work shows that $\mathcal{R} \subseteq F(\mathcal{R})$, thus
$$A \times A <: A$$

λ-calculus

typing rules à la [Cardone and Coppo, 1991]

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

An equi-recursive system

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

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

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

Example

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

$$
\dfrac{
  \dfrac{\dfrac{\dfrac{}{x : A \to A \vdash x : A \to A} \qquad \dfrac{\dfrac{}{x : A \to A \vdash x : A \to A}\;(\approx)}{x : A \to A \vdash x : A}}{x : A \to A \vdash xx : (A \to A) \to A}}{x : A \to A \vdash xx : A}\;(\approx)
}{
  \vdash \lambda x.xx : (A \to A) \to A
}
$$

$$
\dfrac{
  \dfrac{\dfrac{\dfrac{\dfrac{}{x : A \vdash x : A}}{x : A \vdash x : A \to ((A \to A) \to A)}\;(\approx) \qquad \dfrac{}{x : A \vdash x : A}\;(\approx)}{x : A \vdash xx : (A \to A) \to A}}{\vdash \lambda x.xx : A \to ((A \to A) \to A)}
  \qquad
  \dfrac{\vdots}{\vdash \lambda x.xx : A}
}{
  \vdash (\lambda x.xx)(\lambda x.xx) : (A \to A) \to A
}
$$

λ-calculus

typing rules à la [Cardone and Coppo, 1991]

An equi-recursive system

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

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

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

- Strong Normalisation is false! $\vdash (\lambda x.(xx))(\lambda x.(xx))$

Pour le TP

Implement

- *treeof*
- decision procedure for $\approx$
- Project: type inference for recursive types have fun!