Typage

Syntax and semantics

2018-2019

Giovanni Bernardi, `gioXYZirif.fr`
`http://www.irif.fr/~gio/index.xhtml`
Université Paris Diderot

Practical information

Teaching

Lectures	wednesdays	2027	09h30 – 11h00
TP	wednesdays	2027	11h00 – 12h30

No lecture/TP 16 Jan!!!

OPCT'19

Final grade

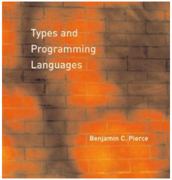
- Exam 40%
- Projet 60%
  1. One `.ml` file that contains ocaml
  2. Type inference + unification algorithms finite types
  3. Type unification algorithm recursive types

Material

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

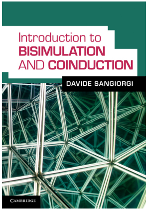
"Types and Programming Languages"
B. Pierce

few chapters



"Introduction To Bisimulation and Coinduction"
D. Sangiori

one chapter



don't be shy :-)

Do ask questions!!!

feedback helps

Overall aim

Make you understand (co)induction. fixed points

Show you different

- standpoints on the subject matter
- applications

First part of the lectures

1. Mini functional language λ-calculus
2. Operational semantics
3. Monomorphic types
4. Polymorphic types

Who introduced λ-calculus ?

An Unsolvable Problem
of Elementary Number Theory

Alonzo Church, 1936

annus mirabilis CS

$\boxed{1936}$

`https://functionaljobs.com/`

A functional language

Syntax

$$
\begin{array}{rlll}
M, N ::= & x & (\mathit{variable}) & | \\
 & ct & (\mathit{constant}) & | \\
 & \langle M, N \rangle & (\mathit{pair}) & | \\
 & M\ N & (\mathit{application}) & | \\
 & \lambda x.M & (\mathit{abstraction}) & | \\
 & \mathbf{let}\ x = M\ \mathbf{in}\ N & (\mathit{let}) &
\end{array}
$$

Some constants :

fst, *snd*, *fix*, *ifthenelse*, $+$, $*$, ..., *true*, *false*, 0, 1, 2, 3, ...

- Do you understand the syntax above ?
- Note circularity: M, N appear left and right of $::=$

questions questions

What does a program do? And why ?

Notations

$$
\begin{array}{lcl}
M_1 M_2 \dots M_n & \equiv & (\dots((M_1\ M_2)M_3)\dots M_{n-1})\ M_n \\
N\ \vec{M} & \equiv & (\dots(((N\ M_1)\ M_2)M_3)\dots M_{n-1})\ M_n \\
M + N & \equiv & +\langle M, N\rangle \\
\textbf{if } E \textbf{ then } M \textbf{ else } N & \equiv & \mathit{ifthenelse}\langle E, \langle M, N\rangle\rangle
\end{array}
$$

Free variables

$$
\begin{array}{lcl}
FV(x) & = & \{x\} \\
FV(ct) & = & \emptyset \\
FV(\langle M, N \rangle) & = & FV(M) \cup FV(N) \\
FV(M\ N) & = & FV(M) \cup FV(N) \\
FV(\lambda x.M) & = & FV(M) \setminus \{x\} \\
FV(\textbf{let}\ x = M\ \textbf{in}\ N) & = & FV(M) \cup FV(N) \setminus \{x\}
\end{array}
$$

A variable x is free in M if $x \in FV(M)$.
A term M is closed iff it has no free variable, i.e. $FV(M) = \emptyset$.

Example

- $M = \lambda z.((\lambda x.x\ z)(\lambda y.y))$ is closed $\qquad FV(M) = \emptyset$
- $M = (\lambda x.x\ z)(\lambda y.y)$ is not closed $\qquad FV(M) = \{z\}$

Note circularity: FV appears left and right of $=$

Bound variables

$$
\begin{array}{lcl}
BV(x) & = & \emptyset \\
BV(ct) & = & \emptyset \\
BV(\langle M, N \rangle) & = & BV(M) \cup BV(N) \\
BV(M\ N) & = & BV(M) \cup BV(N) \\
BV(\lambda x.M) & = & BV(M) \cup \{x\} \\
BV(\textbf{let}\ x = M\ \textbf{in}\ N) & = & BV(M) \cup BV(N) \cup \{x\}
\end{array}
$$

A variable x is bound in M if $x \in BV(M)$.

Example

- x not bound in x
- x bound in $\lambda x.x$
- x free and bound in $x\ (\lambda x.x)$

Note circularity: BV appears left and right of $=$

Alpha-conversion

$=_\alpha$ equivalence allowing renaming bound variables. informal definition

Example

- $x\ (\lambda x.x\ y) =_\alpha x\ (\lambda z.z\ y)$
- $\textbf{let}\ x = x'\ \textbf{in}\ x\ y =_\alpha \textbf{let}\ z = x'\ \textbf{in}\ z\ y$
- more in general $\int x^2 dx =_\alpha \int y^2 dy$

Theorem

For every term M there is a term M' such that

1. $M =_\alpha M'$
2. **Barendregt's Convention**:
   - $FV(M') \cap BV(M') = \emptyset$.
   - *All the bound variables of M' are distinct.*

Substitution

The application of a substitution $\sigma = \{x_1/M_1, \ldots, x_n/M_n\}$ to a term M is defined by induction[1] as follows:

$$
\begin{array}{lcll}
\sigma x_i & = & M_i & \text{If } i \in \{1, \ldots, n\} \\
\sigma y & = & y & \text{If } y \notin \{x_1, \ldots, x_n\} \\
\sigma ct & = & ct & \\
\sigma \langle M, N \rangle & = & \langle \sigma M, \sigma N \rangle & \\
\sigma (M\ N) & = & \sigma M\ \sigma N & \\
\sigma (\lambda x.M) & = & \lambda x.(\sigma M) & \text{If no capture of variables} \\
\sigma (\textbf{let } x = M \textbf{ in } N) & = & \textbf{let } x = \sigma M \textbf{ in } \sigma N & \text{If no capture of variables}
\end{array}
$$

Example

$\{y/x\}(\lambda x.y\ x) = \lambda z.(\{y/x\}x\ z) = \lambda z.x\ z$

[1] Note circularity: σ appears left and right of $=$

Operational Semantics

Defining an Operational Semantics

- Granularity
  - Big-step
  - Small-step
- Order of evaluation
  - Call-by-value
  - Call-by-name
  - ...
- Inference rules + derivation trees

inference rule

$$\boxed{\dfrac{\text{premise}_1 \quad \dots \quad \text{premise}_n}{\text{conclusion}} \ \text{side condition}}$$

Big-step Semantics

Each rule[2] completely evaluates the expression to a value.

Sketch

- a arithmetic expression, σ state, n value
- in state σ the expression a evaluates to n $\qquad (a, \sigma) \Downarrow n$

$$\frac{}{(n, \sigma) \Downarrow n} \qquad \frac{}{(X, \sigma) \Downarrow \sigma(X)}$$

$$\frac{(a_1, \sigma) \Downarrow n_1 \qquad (a_2, \sigma) \Downarrow n_2}{(a_1 + a_2, \sigma) \Downarrow n} \; n \text{ is "} n_1 \text{ plus } n_2 \text{"}$$

We write $(a, \sigma) \Downarrow n$ if there exists **finite** derivation tree $\dfrac{\vdots}{(a, \sigma) \Downarrow n}$

[2] Note circularity: $\Downarrow$ appears in premises and conclusions of rules

Properties

- Abstract
- Allows to avoid details
- No specification of evaluation order (e.g. $(1+3)+(5-3)$)
- No specification of control of errors
- No specification of interleaving

Who introduced small-step semantics ?

A Structural Approach to Operational Semantics

Gordon Plotkin, 1981

It is the purpose of these notes to develop a simple and direct method for specifying the semantics of programming languages. Very little is required in the way of mathematical background; all that will be involved is "symbol-pushing" [...]

Small-step Semantics

Evaluation is sequence of *state changes* of an abstract machine which terminates when the state cannot be reduced further.

Sketch[3]

$$\frac{}{(X, \sigma) \rightsquigarrow (\sigma(X), \sigma)} \qquad \frac{}{(n_1 + n_2, \sigma) \rightsquigarrow (n, \sigma)}\ n \text{ is ``}n_1 \text{ plus } n_2\text{''}$$

$$\frac{(a_1, \sigma) \rightsquigarrow (a_1', \sigma')}{(a_1 + a_2, \sigma) \rightsquigarrow (a_1' + a_2, \sigma')} \qquad \frac{(a_2, \sigma) \rightsquigarrow (a_2', \sigma')}{(n_1 + a_2, \sigma) \rightsquigarrow (n_1 + a_2', \sigma')}$$

We write $(a, \sigma) \rightsquigarrow (a', \sigma')$ if there exists **finite** derivation tree $\frac{\vdots}{(a, \sigma) \rightsquigarrow (a', \sigma')}$

[3] Note circularity: $\rightsquigarrow$ appears in premises and conclusions of rules

Properties

- Less abstract
- Specification of order of evaluation
- Control of errors: $\dfrac{n_2 \neq 0}{n_1 / n_2 \rightsquigarrow n}$, where n is "n_1 divided by n_2".
- Interleaving: $\dfrac{\langle c_1, \sigma \rangle \rightsquigarrow \langle c_1', \sigma' \rangle}{\langle c_1 || c_2, \sigma \rangle \rightsquigarrow \langle c_1' || c_2, \sigma' \rangle}$

From Small-step to Multi-step Semantics

The multi-step semantics is given by the relation $t \rightsquigarrow^* t'$ which is the reflexive and transitive closure of $t \rightsquigarrow t'$.

(P1) $t \rightsquigarrow^* t$ for every t

(P2) $t \rightsquigarrow t'$ implies $t \rightsquigarrow^* t'$

(P3) $t \rightsquigarrow^* t'$ and $t' \rightsquigarrow^* t''$ implies $t \rightsquigarrow^* t''$

Properties of the small and big step semantics

- The relation $\rightsquigarrow$ is deterministic.
- The relation $\Downarrow$ is deterministic.
- $t \Downarrow v$ iff $t \rightsquigarrow^* v$, where v is a "value".

Normal Forms

- A normal form is a term that cannot be evaluated any further: is a state where the abstract machine is halted (result of the evaluation).

Big-step versus small-step semantics

- In small-step semantics evaluation stops at errors. In big-step semantics errors occur deeply inside derivation trees.
- The order of evaluation is *explicit* in small-step semantics but *implicit* in big-step semantics.
- Big-step semantics is more abstract, but less precise.
- Small-step semantics allows to make difference between non-termination and "getting stuck".

Reduction Rules

$$
\begin{array}{lcll}
(\lambda x.M)\ N & \rightarrow & M\{x/N\} & \beta\text{-reduction} \\
\textbf{let } x = N \textbf{ in } M & \rightarrow & M\{x/N\} & \\
\mathit{fix}\ M & \rightarrow & M\ (\mathit{fix}\ M) & \\
\mathit{fst}\langle M, N\rangle & \rightarrow & M & \\
\mathit{snd}\langle M, N\rangle & \rightarrow & N & \\
\textbf{if } \mathit{true} \textbf{ then } M \textbf{ else } N & \rightarrow & M & \\
\textbf{if } \mathit{false} \textbf{ then } M \textbf{ else } N & \rightarrow & N & \\
\textbf{if } 0 \textbf{ then } M \textbf{ else } N & \rightarrow & M & \\
\textbf{if } n \textbf{ then } M \textbf{ else } N & \rightarrow & N, \quad n \neq 0 &
\end{array}
$$

WARNING!: The reduction relation $\rightarrow$ is non-deterministic.

Call-by-value lambda-calculus (big-step semantics)

$$(\textbf{Values})\ V ::= ct \mid \langle V, V \rangle \mid \lambda x.M \mid \mathit{fix}\ \ M$$

Meaningless expressions such as $(\langle 1, 1 \rangle\ 3)$ or $(\mathit{true}\ 3)$ are not considered as values.

$$\frac{}{V \ \Downarrow_v \ V}\ \text{V is a value} \qquad \frac{M_1 \Downarrow_v V_1 \qquad M_2 \Downarrow_v V_2}{\langle M_1, M_2 \rangle \Downarrow_v \langle V_1, V_2 \rangle}$$

$$\frac{M \Downarrow_v \lambda x.L \qquad N \Downarrow_v W \qquad L\{x/W\} \Downarrow_v V}{M\ N \Downarrow_v V}$$

$$\frac{N \Downarrow_v V \qquad L\{x/V\} \Downarrow_v W}{\textbf{let}\ x = N\ \textbf{in}\ L \Downarrow_v W}$$

$$\frac{M \Downarrow_v \mathit{fix}\ L \qquad N \Downarrow_v W \qquad (L\ (\mathit{fix}\ L))\ W \Downarrow_v V}{M\ N \Downarrow_v V}$$

$$\frac{M \Downarrow_v \mathit{fst} \qquad N \Downarrow_v \langle V_1, V_2 \rangle}{M\ N \Downarrow_v V_1} \qquad \frac{M \Downarrow_v \mathit{snd} \qquad N \Downarrow_v \langle V_1, V_2 \rangle}{M\ N \Downarrow_v V_2}$$

$$\frac{M \Downarrow_v \mathit{true} \qquad N \Downarrow_v V}{\textbf{if } M \textbf{ then } N \textbf{ else } L \Downarrow_v V} \qquad \frac{M \Downarrow_v \mathit{false} \qquad L \Downarrow_v V}{\textbf{if } M \textbf{ then } N \textbf{ else } L \Downarrow_v V}$$

$$\frac{M \Downarrow_v 0 \qquad N \Downarrow_v V}{\textbf{if } M \textbf{ then } N \textbf{ else } L \Downarrow_v V} \qquad \frac{M \Downarrow_v n \qquad n \neq 0 \qquad L \Downarrow_v V}{\textbf{if } M \textbf{ then } N \textbf{ else } L \Downarrow_v V}$$

Particular case: closed pure lambda-terms

$$(\textbf{Values})\ V ::= \lambda x.M$$

$$\frac{}{V \Downarrow_v V} \qquad \frac{M \Downarrow_v \lambda x.L \quad N \Downarrow_v W \quad L\{x/W\} \Downarrow_v V}{M\ N \Downarrow_v V}$$

An example

$M = \lambda f.\lambda x.\langle x, f\ x\rangle$ and $N = \lambda y.y$.

$$\frac{M \Downarrow_v M \qquad N \Downarrow_v N \qquad \lambda x.\langle x, f\ x\rangle\{f/N\} \Downarrow_v \lambda x.\langle x, N\ x\rangle}{M\ N \Downarrow_v \lambda x.\langle x, N\ x\rangle}$$

$$\cfrac{\cfrac{}{M\ N \Downarrow_v \lambda x.\langle x, N\ x\rangle} \vdots 1 \Downarrow_v 1 \qquad \cfrac{1 \Downarrow_v 1 \qquad \cfrac{N \Downarrow_v N \qquad 1 \Downarrow_v 1 \qquad y\{y/1\} \Downarrow_v 1}{N\ 1 \Downarrow_v 1}}{\langle 1, N\ 1\rangle \Downarrow_v \langle 1, 1\rangle}}{M\ N\ 1 \Downarrow_v \langle 1, 1\rangle}$$

Call-by-value lambda calculus (small-step semantics)

$$\frac{M \leadsto_v M'}{M\ N \leadsto_v M'\ N} \qquad \frac{N \leadsto_v N'}{V\ N \leadsto_v V\ N'}$$

$$\frac{}{(\lambda x.M)\ V \leadsto_v M\{x/V\}} \qquad \frac{}{(\mathit{fix}\ M)\ V \leadsto_v (M\ (\mathit{fix}\ M))\ V}$$

$$\frac{N \leadsto_v N'}{\mathbf{let}\ x = N\ \mathbf{in}\ L \leadsto_v \mathbf{let}\ x = N'\ \mathbf{in}\ L} \qquad \frac{}{\mathbf{let}\ x = V\ \mathbf{in}\ L \leadsto_v L\{x/V\}}$$

$$\frac{M \leadsto_v M'}{\langle M, N\rangle \leadsto_v \langle M', N\rangle} \qquad \frac{N \leadsto_v N'}{\langle V, N\rangle \leadsto_v \langle V, N'\rangle}$$

$$\frac{}{\mathit{fst}\ \langle V_1, V_2\rangle \leadsto_v V_1} \qquad \frac{}{\mathit{snd}\ \langle V_1, V_2\rangle \leadsto_v V_2}$$

$$\frac{M \rightsquigarrow_v M'}{\textbf{if } M \textbf{ then } N \textbf{ else } L \rightsquigarrow_v \textbf{if } M' \textbf{ then } N \textbf{ else } L}$$

$$\frac{}{\textbf{if } \mathit{true} \textbf{ then } N \textbf{ else } L \rightsquigarrow_v N} \qquad \frac{}{\textbf{if } \mathit{false} \textbf{ then } N \textbf{ else } L \rightsquigarrow_v L}$$

$$\frac{}{\textbf{if } 0 \textbf{ then } N \textbf{ else } L \rightsquigarrow_v N} \qquad \frac{n \neq 0}{\textbf{if } n \textbf{ then } N \textbf{ else } L \rightsquigarrow_v L}$$

The same example

Small-step semantics

Let $M = \lambda f.\lambda x.\langle x, f\ x\rangle$ and $N = \lambda y.y$.

$$\begin{array}{ll} M\ N\ 1 & \rightsquigarrow_v \\ (\lambda x.\langle x, N\ x\rangle)\ 1 & \rightsquigarrow_v \\ \langle 1, N\ 1\rangle & \rightsquigarrow_v \\ \langle 1, 1\rangle & \end{array}$$

Call-by-name lambda-calculus (big-step semantics)

$$(\textbf{Lazy Forms})\ P ::= ct \mid \langle M, N\rangle \mid \lambda x.M \mid \mathit{fix}\ M$$

$$\frac{M \Downarrow_n \lambda x.L \qquad L\{x/N\} \Downarrow_n P}{M\ N \Downarrow_n P} \qquad \frac{P \text{ is a lazy form}}{P \Downarrow_n P}$$

$$\frac{L\{x/N\} \Downarrow_n P}{\textbf{let } x = N \textbf{ in } L \Downarrow_n P} \qquad \frac{M \Downarrow_n \mathit{fix}\ L \qquad (L\ (\mathit{fix}\ L))\ N \Downarrow_n P}{M\ N \Downarrow_n P}$$

$$\frac{M \Downarrow_n \langle M_1, M_2\rangle \qquad M_1 \Downarrow_n P_1}{\mathit{fst}\ M \Downarrow_n P_1} \qquad \frac{M \Downarrow_n \langle M_1, M_2\rangle \qquad M_2 \Downarrow_n P_2}{\mathit{snd}\ M \Downarrow_n P_2}$$

$$\frac{M \Downarrow_n \mathit{true} \qquad N \Downarrow_n P}{\textbf{if } M \textbf{ then } N \textbf{ else } L \Downarrow_n P} \qquad \frac{M \Downarrow_n \mathit{false} \qquad L \Downarrow_n P}{\textbf{if } M \textbf{ then } N \textbf{ else } L \Downarrow_n P}$$

$$\frac{M \Downarrow_n 0 \qquad N \Downarrow_n P}{\textbf{if } M \textbf{ then } N \textbf{ else } L \Downarrow_n P} \qquad \frac{M \Downarrow_n n \qquad n \neq 0 \qquad L \Downarrow_n P}{\textbf{if } M \textbf{ then } N \textbf{ else } L \Downarrow_n P}$$

Particular case: closed pure lambda-terms

$$(\textbf{Lazy Forms})\ P ::= \lambda x.M$$

$$\frac{}{P \ \Downarrow_n \ P} \qquad \frac{M \Downarrow_n \lambda x.L \qquad L\{x/N\} \Downarrow_n P}{M\ N \Downarrow_n P}$$

An example

Let $M = \lambda f.\lambda x.\langle x, (f\ x)\rangle$ and $M_f = \mathit{fix}\ M$.

$$\dfrac{\dfrac{}{\mathit{fix}\ M \Downarrow_n \mathit{fix}\ M} \qquad \dfrac{\dfrac{\vdots}{M\ M_f \Downarrow_n \lambda x.\langle x, M_f\ x\rangle} \qquad \dfrac{}{\langle x, M_f\ x\rangle\{x/1\} \Downarrow_n \langle 1, M_f\ 1\rangle}}{M\ (\mathit{fix}\ M)\ 1 \Downarrow_n \langle 1, \mathit{fix}\ M\ 1\rangle}}{\mathit{fix}\ M\ 1 \Downarrow_n \langle 1, \mathit{fix}\ M\ 1\rangle}$$

$$\dfrac{\dfrac{}{M \Downarrow_n M} \qquad (\lambda x.\langle x, f\ x\rangle)\{f/M_f\} \Downarrow_n \lambda x.\langle x, M_f\ x\rangle}{M\ M_f \Downarrow_n \lambda x.\langle x, M_f\ x\rangle}$$

Exercice

Try to evaluate $\mathit{fix}\ M\ 1 \Downarrow_v$.

Call-by-name lambda calculus (small-step semantics)

$$\frac{M \rightsquigarrow_n M'}{M\ N \rightsquigarrow_n M'\ N}$$

$$\frac{}{(\lambda x.M)\ N \rightsquigarrow_n M\{x/N\}} \qquad \frac{}{(\mathit{fix}\ M)\ N \rightsquigarrow_n (M\ (\mathit{fix}\ M))\ N}$$

$$\frac{}{\mathbf{let}\ x = M\ \mathbf{in}\ L \rightsquigarrow_n L\{x/M\}}$$

$$\frac{M \rightsquigarrow_n M'}{\mathit{fst}\ M \rightsquigarrow_n \mathit{fst}\ M'} \qquad \frac{}{\mathit{fst}\ \langle M, N\rangle \rightsquigarrow_n M}$$

$$\frac{M \rightsquigarrow_n M'}{\mathit{snd}\ M \rightsquigarrow_n \mathit{snd}\ M'} \qquad \frac{}{\mathit{snd}\ \langle M, N\rangle \rightsquigarrow_n N}$$

$$\frac{M \rightsquigarrow_n M'}{\textbf{if } M \textbf{ then } N \textbf{ else } L \rightsquigarrow_n \textbf{if } M' \textbf{ then } N \textbf{ else } L}$$

$$\frac{}{\textbf{if } \mathit{true} \textbf{ then } N \textbf{ else } L \rightsquigarrow_n N} \qquad \frac{}{\textbf{if } \mathit{false} \textbf{ then } N \textbf{ else } L \rightsquigarrow_n L}$$

$$\frac{}{\textbf{if } 0 \textbf{ then } N \textbf{ else } L \rightsquigarrow_n N} \qquad \frac{n \neq 0}{\textbf{if } n \textbf{ then } N \textbf{ else } L \rightsquigarrow_n L}$$

The same example

Small-step semantics

Let $M = \lambda f.\lambda x.\langle x, (f\ x)\rangle$.

$$\begin{array}{ll} \mathit{fix}\ M\ 1 & \rightsquigarrow_n \\ M\ (\mathit{fix}\ M)\ 1 & \rightsquigarrow_n \\ (\lambda x.\langle x, (\mathit{fix}\ M\ x)\rangle)\ 1 & \rightsquigarrow_n \\ \langle 1, (\mathit{fix}\ M\ 1)\rangle & \end{array}$$

Coherence of results

- If $M \Downarrow_v N$, then N is a value.
- If $M \Downarrow_n N$, then N is a lazy form.

Deterministic properties

- If $M \Downarrow_v V$ and $M \Downarrow_v V'$, then $V = V'$.
- If $M \Downarrow_n P$ and $M \Downarrow_n P'$, then $P = P'$.
- If $M \leadsto_v N$ and $M \leadsto_v N'$, then $N = N'$.
- If $M \leadsto_n N$ and $M \leadsto_n N'$, then $N = N'$.

Relating big and small-steps semantics

- If $M \Downarrow_v V$, then $M \leadsto_v^* V$.
- If $M \Downarrow_n P$, then $M \leadsto_n^* P$.
- If $M \leadsto_v^* N$ and N is a value, then $M \Downarrow_v N$.
- If $M \leadsto_n^* N$ and N is a lazy form, then $M \Downarrow_n N$.

Problem

Try evaluating

- $1 + \mathit{true}$
- $\textbf{if } \lambda x.x \textbf{ then } \lambda x.x \textbf{ else } \lambda x.\lambda y.x\ y$