Master ATIAM

Fondamentaux pour l'informatique appliquée à la musique

COURS 3 : Programmation fonctionnelle et par objets

Carlos Agon
agonc@ircam.fr

Démo OM

Lambda-Calcul

Lexique

$$\Sigma = \{\ (,\ )\ ,\ \lambda,\ a,\ b,\ c\ldots\}$$

Lambda-Calcul

Syntaxe

E::= x

E::= E E (Application)

E::= λxE (Abstraction)

Occurrences libres d'une variable

$$OL(x, E)$$

Si E est x $\quad OL(x,E) = x$

Si E est E_1E_2 $\quad OL(x,E) = OL(x, E_1) \cup OL(x, E_2)$

Si E est $\lambda y E'$ $\quad OL(x,E) = OL(x, E') \text{ si } x \neq y$

sinon $\varnothing$

La substitution

$\lambda x.\ xy$

$\lambda x.\ xy[x/y]$ $\cancel{\lambda x.\ xx}$

$\lambda z.\ zy\ [x/y]$ $\lambda z.zx$

$$\lambda x M \equiv \lambda x' M [x/x'] \quad \text{si } x \notin FV(M)$$

La α-réduction

Si y n'est pas libre dans λx.X alors

$$\lambda x.X \quad \rightarrow_{\alpha} \quad \lambda x.X\,[y/x]$$

E $\cong\alpha$ E'

réflexive $\quad$ E $\cong\alpha$ E

symétrique $\quad$ E $\cong\alpha$ E' $\Rightarrow$ E $\cong\alpha$ E'

transitive $\quad$ E $\cong\alpha$ E' , E' $\cong\alpha$ E'' $\Rightarrow$ E $\cong\alpha$ E''

La β-réduction

Redex $(\lambda x.M)\ N$

Forme normale E tq. E n'a pas de redex

$$(\lambda x.M)\ N \quad \rightarrow_\beta \quad M[N/x]$$

$$((\lambda x.\ \lambda y.(x)y)b)c \quad \rightarrow_\beta \quad (\lambda y.(b)y)c$$

$$\rightarrow_\beta \quad (b)c$$

La terminaison

$$(\lambda \mathbf{x}.(\mathbf{x})\mathbf{x})\ \lambda \mathbf{x}.(\mathbf{x})\mathbf{x}$$

$$\to_\beta \quad (\lambda \mathbf{x}.(\mathbf{x})\mathbf{x})\ \lambda \mathbf{x}.(\mathbf{x})\mathbf{x}$$

$$\to_\beta \quad (\lambda \mathbf{x}.(\mathbf{x})\mathbf{x})\ \lambda \mathbf{x}.(\mathbf{x})\mathbf{x}$$

...

$$\to_\beta \quad (\lambda \mathbf{x}.(\mathbf{x})\mathbf{x})\ \lambda \mathbf{x}.(\mathbf{x})\mathbf{x}$$

...

Church Rossel

$$
\begin{tikzcd}
 & E \arrow[dl, "\beta"', "*"] \arrow[dr, "*", "\beta"'] & \\
E_2 \arrow[dr, "*", "\beta"'] & & E_1 \arrow[dl, "\beta", "*"'] \\
 & E_3 &
\end{tikzcd}
$$

La forme normale de E, si elle existe, est unique

Stratégies d'évaluation

$$(\lambda y.v)((\lambda x.(x)x)\ \lambda x.(x)x)$$

$$\to_\beta \quad (\lambda y.v)(\lambda x.(x)x)\ \lambda x.(x)x$$

$$\to_\beta \quad (\lambda y.v)(\lambda x.(x)x)\ \lambda x.(x)x$$

...

$$(\lambda y.v)((\lambda x.(x)x)\ \lambda x.(x)x)$$

$$\to_\beta \quad v$$

Appel par nom

Réduire tjrs le *redex* le plus à gauche

$$\underline{\lambda v.(\lambda z.z)}((\underline{\lambda w.w)(x(\lambda y.y)))}$$

$$\lambda x.z \text{ (fact 10)} \quad \rightarrow_\beta \quad z$$

$$\lambda x.x + x \text{ (fact 10)} \quad \rightarrow_\beta \quad \text{(fact 10) + (fact 10)}$$

Appel par value

Réduire tjrs le *redex* le plus a gauche, mais si l'argument du *redex* est une valeur

$$\lambda v.(\lambda z.z)(\underline{(\lambda w.w)(x\ (\lambda y.y))})$$

$$\lambda x.x + x\ (\text{fact } 10) \quad \rightarrow_\beta \quad \lambda x.x + x\ 3628800$$

$$\rightarrow_\beta \quad 7257600$$

Nombres de Church

$0 = \lambda x.\ \lambda y.y$

$1 = \lambda x.\ \lambda y.(x)y$

$2 = \lambda x.\ \lambda y.(x)(x)\ y$

...

$n = \lambda x.\ \lambda y.\underbrace{(x)(x)\ldots(x)(x)}_{n}\ y$

Suc $= \lambda x.\ \lambda y.\ \lambda z.(y)((x)\ y)\ z$

$$(\lambda x.\ \lambda y.\ \lambda z.(y)((x)\ y)\ z) \qquad \lambda x.\ \lambda y.\underbrace{(x)(x)\ldots(x)(x)}_{n}y$$

$$\rightarrow_\beta \quad (\lambda y.\ \lambda z.(y)((\lambda x.\ \lambda y.\underbrace{(x)(x)\ldots(x)(x)}_{n}y)\ y)\ z)$$

$$\rightarrow_\alpha \quad (\lambda y.\ \lambda z.(y)((\lambda x.\ \lambda a.\underbrace{(x)(x)\ldots(x)(x)}_{n}a)\ y)\ z)$$

$$\rightarrow_\beta \quad (\lambda y.\ \lambda z.(y)(\lambda a.\underbrace{(y)(y)\ldots(y)(y)}_{n}a)\ z)$$

$$\rightarrow_\beta \quad (\lambda y.\ \lambda z.(y)\underbrace{(y)(y)\ldots(y)(y)}_{n}\ z)$$

$$\rightarrow_\beta \quad (\lambda y.\ \lambda z.\underbrace{(y)(y)(y)\ldots(y)(y)}_{n+1}\ z)$$

$\textbf{ADD} = \lambda x.\ \lambda y.\ \lambda a.\ \lambda b.((x)\ a)\ ((y)\ a)\ b$

$((\lambda x.\ \lambda y.\ \lambda a.\ \lambda b.((x)\ a)\ ((y)\ a)\ b) \quad \lambda x.\ \lambda y.(x)(x)\ y \quad \lambda x.\ \lambda y.(x)(x)(x)y$

$\rightarrow_\beta \quad (\lambda y.\ \lambda a.\ \lambda b.((\lambda x.\ \lambda y.(x)(x)\ y)\ a)\ ((y)\ a)\ b)\ \lambda x.\ \lambda y.(x)(x)(x)y$

$\rightarrow_\beta \quad (\lambda a.\ \lambda b.((\lambda x.\ \lambda y.(x)(x)\ y)\ a)\ ((\lambda x.\ \lambda y.(x)(x)(x)y)\ a)\ b)$

$\rightarrow_\beta \quad (\lambda a.\ \lambda b.(\lambda y.(a)(a)\ y)\ (\lambda y.(a)(a)(a)y)\ b)$

$\rightarrow_\beta \quad (\lambda a.\ \lambda b.(a)(a)\ (a)(a)(a)\ b)$

EVALUATION

Faux

$\lambda x\, \lambda y\; y$

Vrai

$\lambda x\, \lambda y\; x$

Et

$\lambda x\, \lambda y((x)\; y)\; x$

1- Prouver que faux et vrai = faux

Récursivité

MULT $=_\beta$ $\lambda x.\ \lambda y.\ \lambda z.(x)(y)z$

IF $=_\beta$ $\lambda x.\ \lambda y.\ \lambda z.((x)y)z$

...

FACT $=_\beta$ $\lambda n.(((\mathbf{IF})(\mathbf{ZERO?})n)\ \mathbf{I})((\mathbf{MULT})\ n)(\mathbf{FACT})\ (\mathbf{PRED})n$

FACT $=_\beta$ $(\mathbf{H})\ \mathbf{FACT}$

où

H $=_\beta$ $\lambda f.\lambda n.(((\mathbf{IF})(\mathbf{ZERO?})n)\ \mathbf{I})((\mathbf{MULT})\ n)(f)\ (\mathbf{PRED})n$

FACT est un point fixe de la fonction **H**

J'ai besoin d'un Y tq.

$\mathbf{Y}(\mathbf{H}) \quad =_\beta \quad \mathbf{FACT} =_\beta \quad (\mathbf{H})\mathbf{FACT}$

$\mathbf{Y} = \lambda h.\ (\lambda x.(h)(x)\ x))\ \lambda x.(h)(x)\ x$

Récursivité

$$\mathtt{Y(E)} \rightarrow_\beta \mathtt{G} \rightarrow_\beta \mathtt{(E)G}$$

$$\mathbf{Y} = \lambda \mathtt{h.}\ (\lambda \mathtt{x.(h)(x)\ x))}\ \lambda \mathtt{x.(h)(x)\ x}$$

$$(\lambda \mathtt{h.}\ (\lambda \mathtt{x.(h)(x)\ x))}\ \lambda \mathtt{x.(h)(x)\ x)\ E}$$

$$\rightarrow_\beta \quad (\lambda \mathtt{x.(E)(x)\ x))}\ \lambda \mathtt{x.(E)(x)\ x} \qquad = \mathtt{G}$$

$$\rightarrow_\beta \quad \mathtt{(E)}\ (\lambda \mathtt{x.(E)(x)\ x)}\ \lambda \mathtt{x.(E)(x)\ x)}$$

$$= \quad \mathtt{(E)\ G}$$

DEMO Py

Programmation par objets

Historique

Rangement des fonctions

« Les années objets »

Musique

La POO

Langage commun de représentation

Avantages

Réification

PPO

Objets = données + opérations

Langages à classes

vs.

Langages par prototypes

Classes

Structures et comportement

Slots

chord

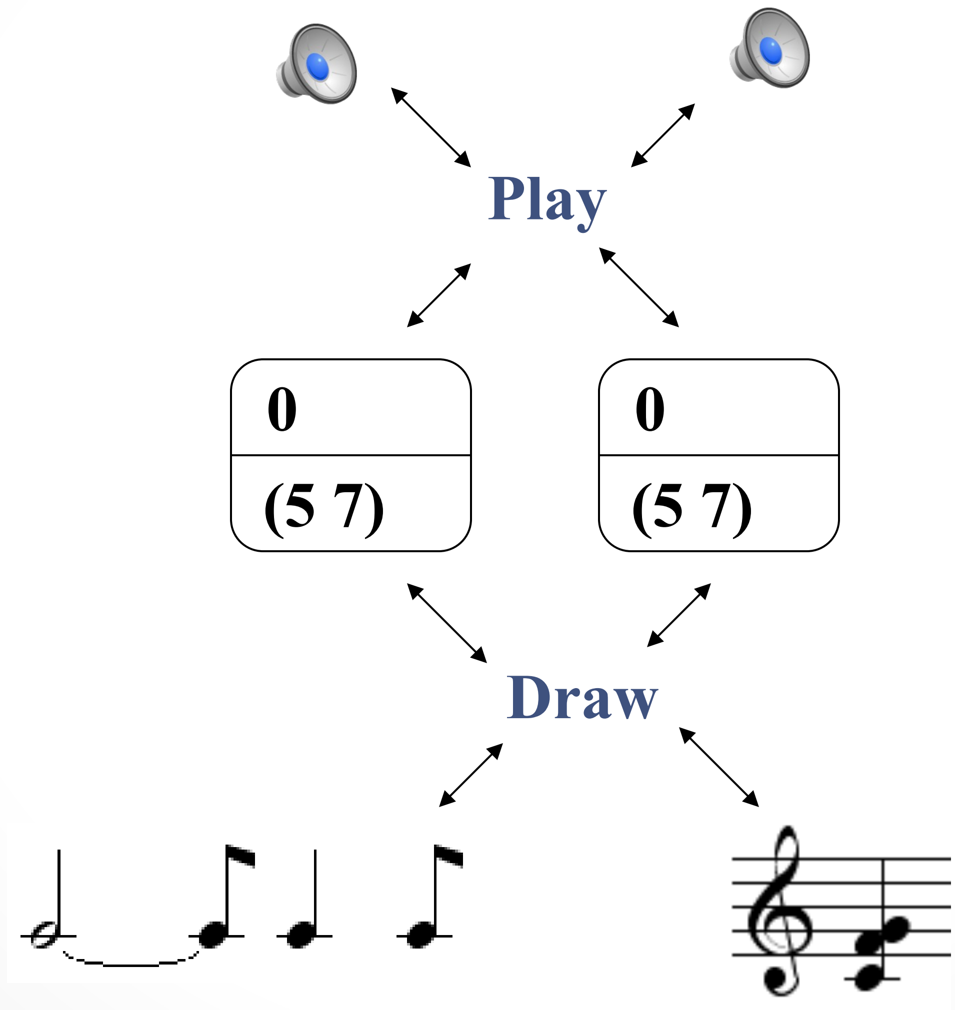
base

intervalle
s

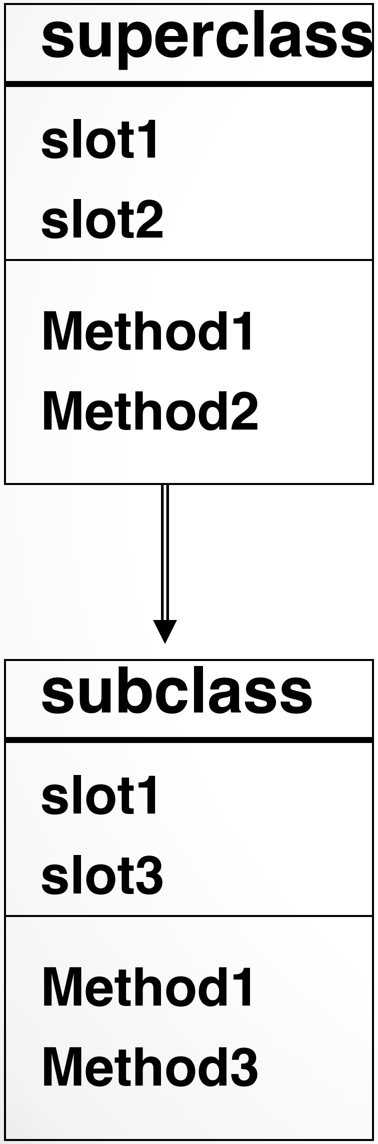
0

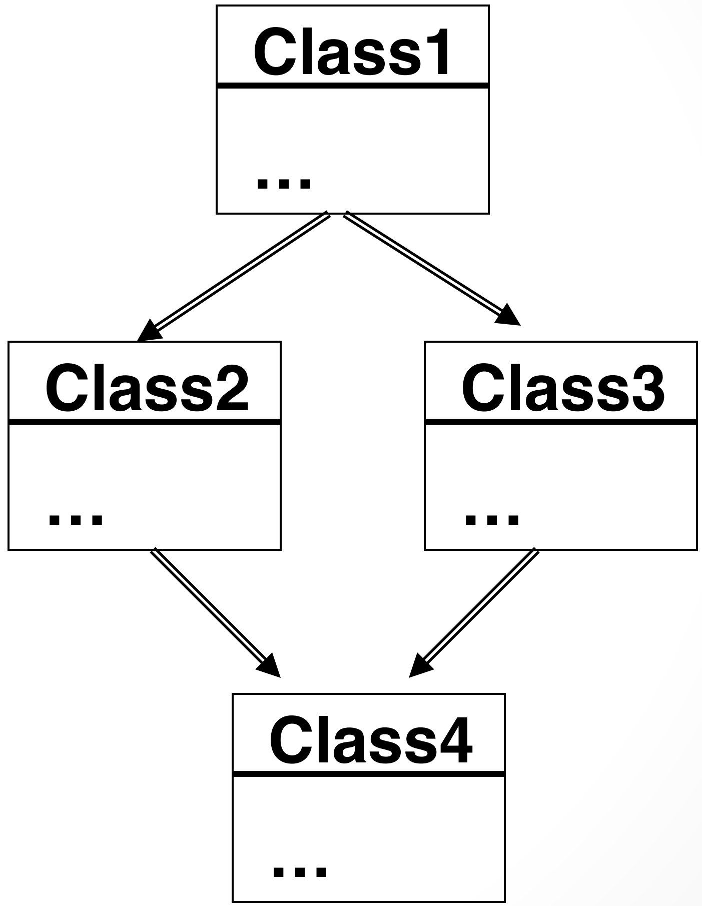
(5 7)

Méthodes

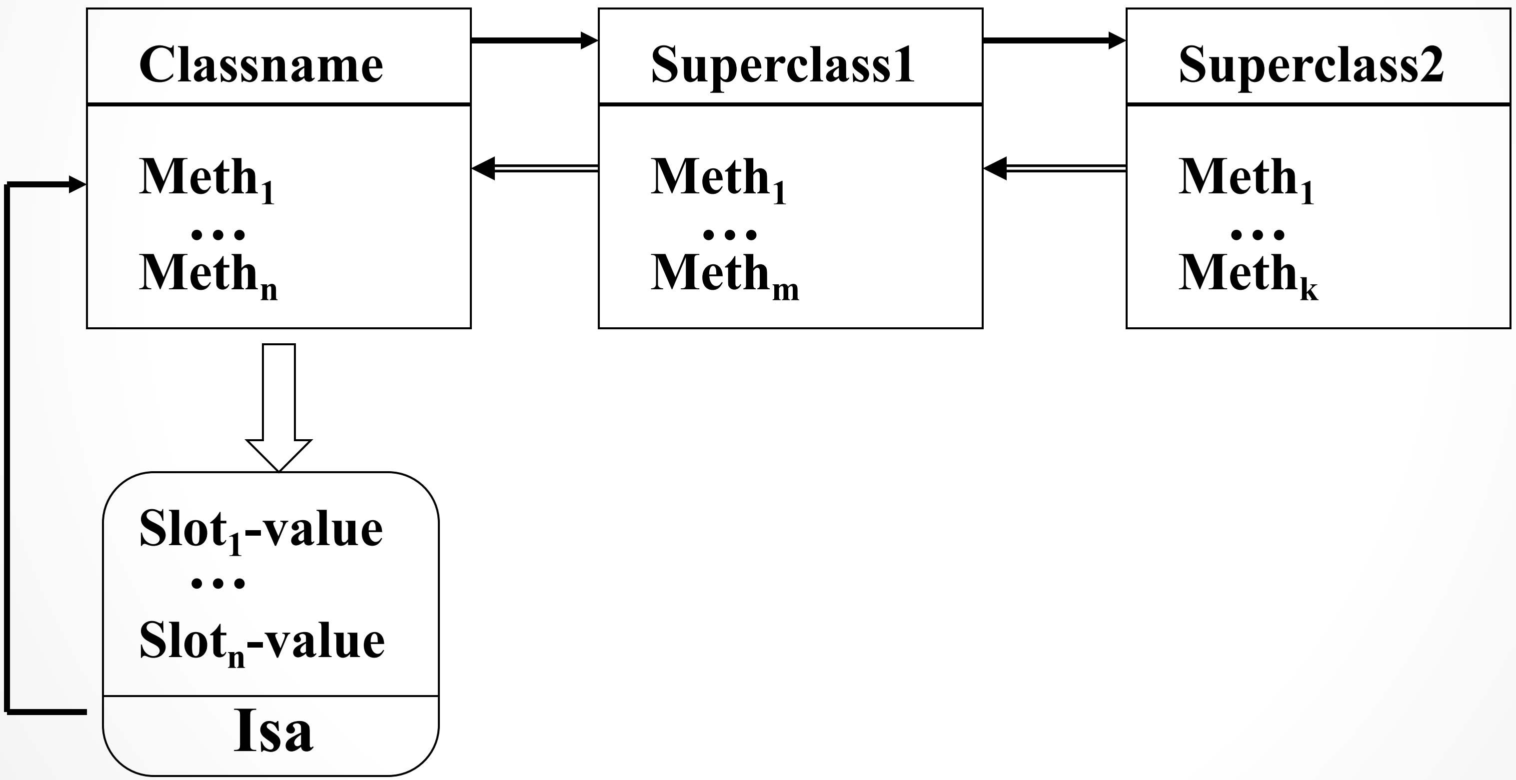


Héritage

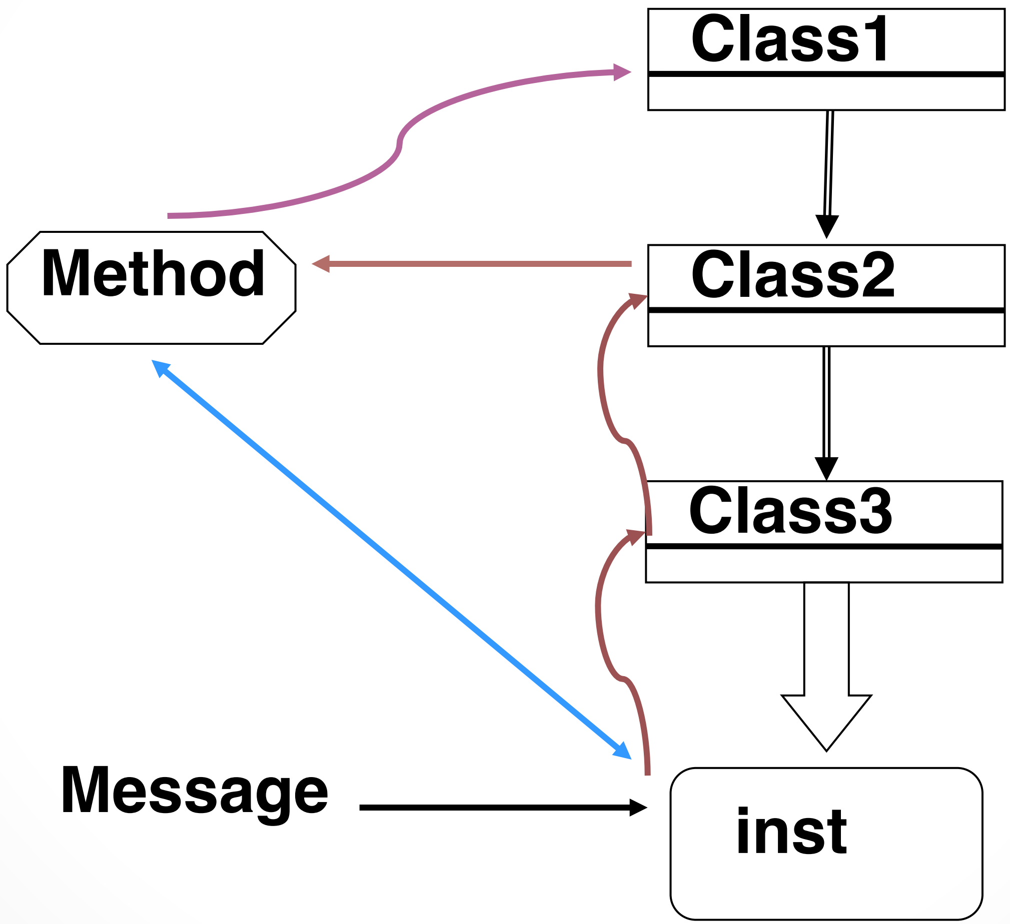

superclass
slot1
slot2
Method1
Method2
subclass
slot1
slot3
Method1
Method3


Class1
...
Class2
...
Class3
...
Class4
...

Instanciation



Dynamic binding



Modélisation

Implémentation