Peut-on classer les langages de programmation ?

Baptiste Mélès

Archives Henri Poincaré, UMR 7117 CNRS, Université de Lorraine (Nancy)

Philosophie de l'informatique, de la logique et de leurs interfaces
(coord. Jean-Baptiste Joinet),
ENS Ulm, lundi 4 mars 2012

Calcul du maximum en langages machine et assembleur

Assembled instructions						Line no.	LOC	OP	ADDRESS	Times	Remarks
						01	X	EQU	1000		
						02		ORIG	3000		
3000:	+	3009	0	2	32	*03*	MAXIMUM	STJ	EXIT	1	Subroutine linkage
3001:	+	0	1	2	51	*04*	INIT	ENT3	0,1	1	*M1. Initialize.* $k \leftarrow n$.
3002:	+	3005	0	0	39	*05*		JMP	CHANGEM	1	$j \leftarrow n$, $m \leftarrow X[n]$, $k \leftarrow n-1$.
3003:	+	1000	3	5	56	*06*	LOOP	CMPA	X,3	$n-1$	*M3. Compare.*
3004:	+	3007	0	7	39	*07*		JGE	*+3	$n-1$	To M5 if $m \geq X[k]$.
3005:	+	0	3	2	50	*08*	CHANGEM	ENT2	0,3	$A+1$	*M4. Change m.* $j \leftarrow k$.
3006:	+	1000	3	5	08	*09*		LDA	X,3	$A+1$	$m \leftarrow X[k]$.
3007:	+	1	0	1	51	*10*		DEC3	1	n	*M5. Decrease k.*
3008:	+	3003	0	2	43	*11*		J3P	LOOP	n	*M2. All tested?* To M3 if $k>0$.
3009:	+	3009	0	0	39	*12*	EXIT	JMP	*	1	Return to main program. ∎

(Knuth 1997, p. 145 : machine `MIX`)

Une factorielle en Lisp

```lisp
(defun fact (x)
  (if (= x 0)
    1
    (* x (fact (- x 1)) ) )
  )
)

(fact 10)
```

Lisp

Une factorielle en C

```c
#include <stdio.h>

void fact (int n) {
  int result = 1;

  for (int i = 1; i <= n; i++) {
    result = result * i;
  }

  printf("%d! = %d\n", n, result);
}

int main (void) {
  int n = 10;
  fact(n);
  return 0;
}
```

```
#include <iostream>

class Number {
public:
  int value;
  void init(int n) { value = n; }
  void fact(void) {
    int result = 1;
    for (int i = 1; i <= value; i++) {
      result = result * i; }
    std::cout << "10! = " << result << "\n";
  }
};

int main (void) {
  Number n;
  n.init(10);
  n.fact();
  return 0;
}
```

Énoncés en Prolog

Codage :

```
/* Faits */
etredans(Nancy, Lorraine).
etredans(Lorraine, France).

/* Regles */
etredans(X,Z) :- etredans(X,Y), etredans(Y,Z).
```

Prolog

Interrogation :

```
?- etredans(Nancy, France).
   true
?-
```

Prolog

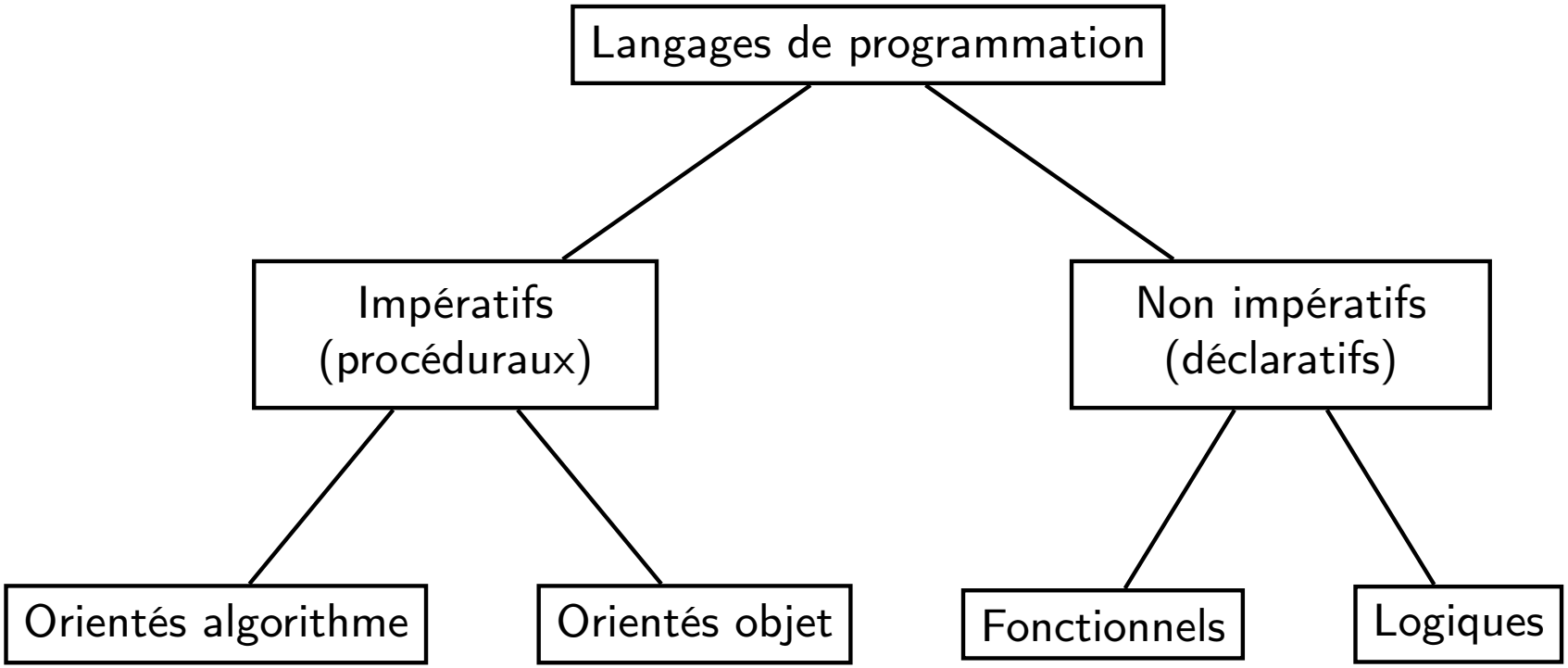
Sucre syntaxique

Formulations équivalentes :

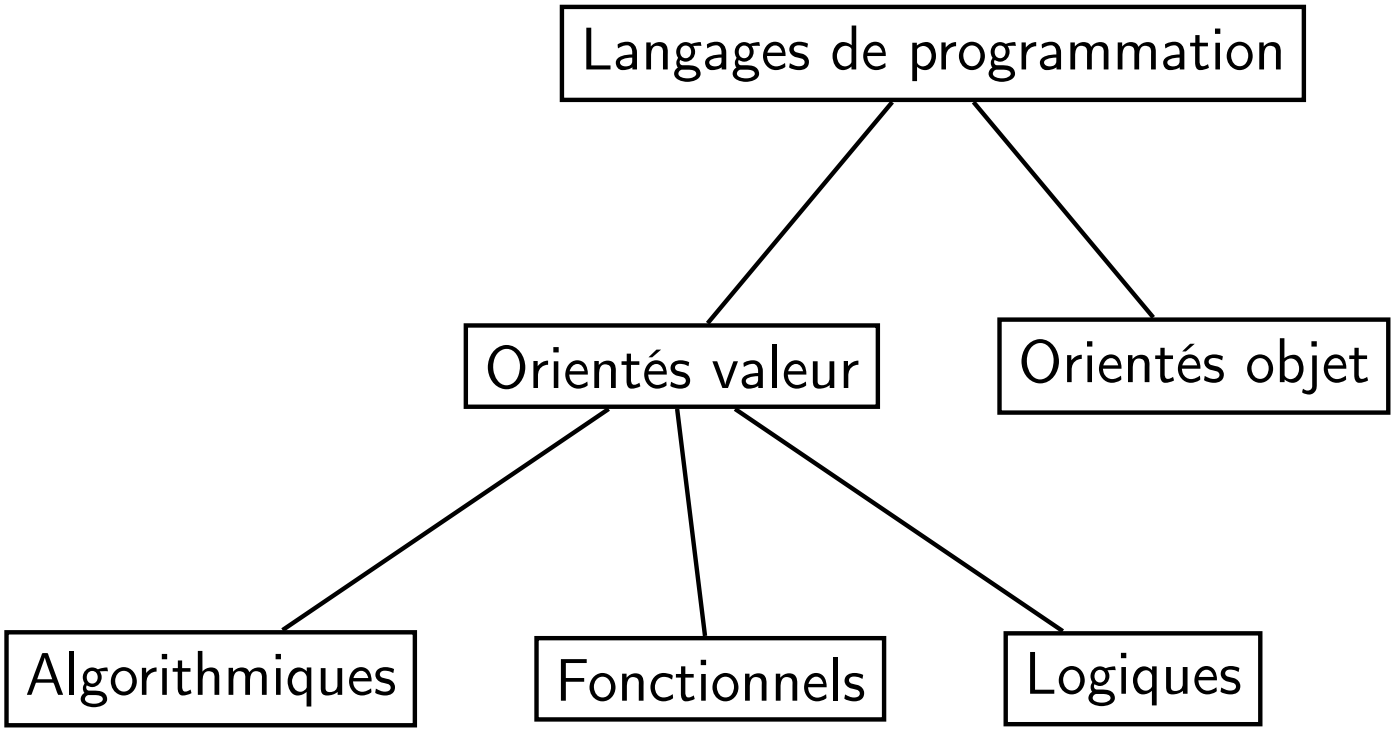
i = 0;
etiquette:
if (i < 10) {
  commandes;
  i = i + 1;
  goto etiquette:
}
	i = 0;
	while (i < 10) {
	  commandes;
	  i = i + 1;
	}
		for (i = 0; i < 10; i = i+1) {
		  commandes;
		}

C

Première classification de Rechenberg (1990)



Seconde classification de Rechenberg (1990)



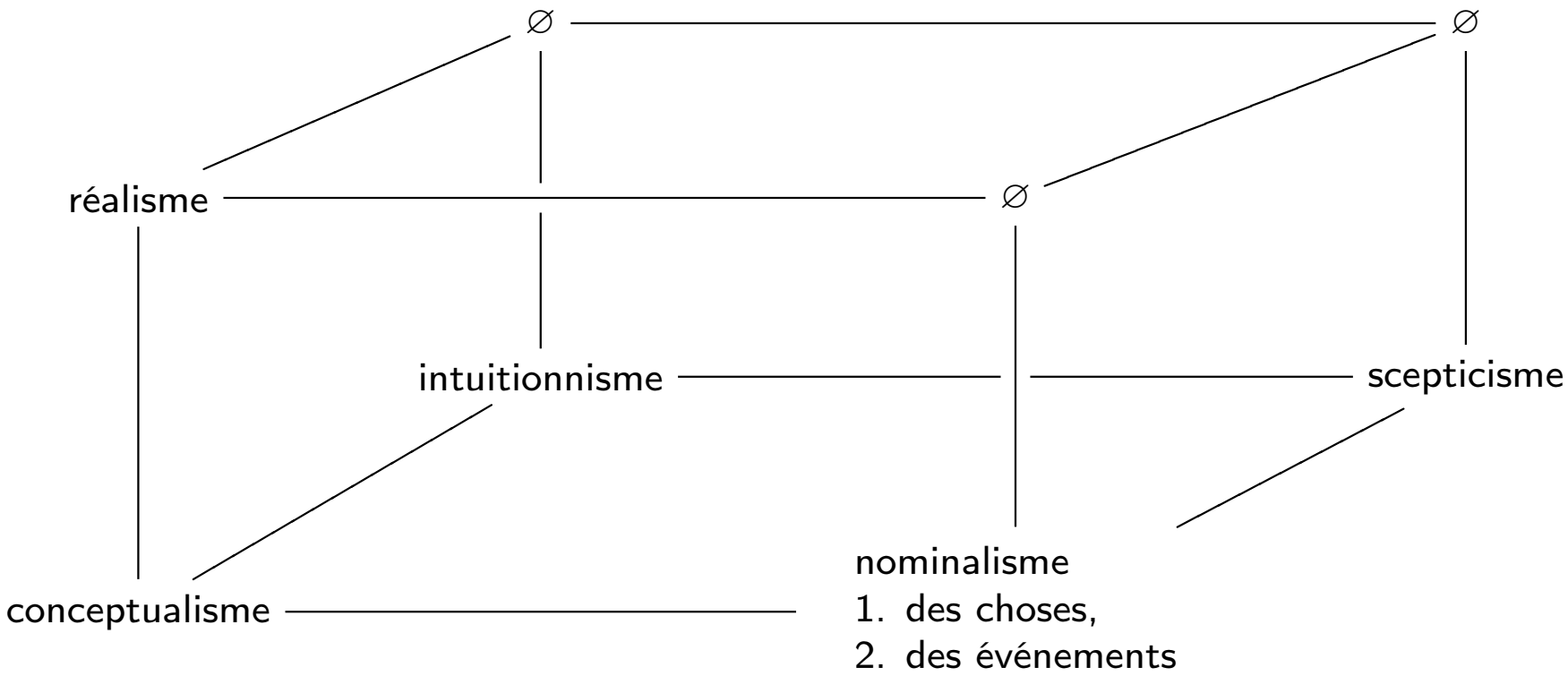
Classification de Jules Vuillemin (1984)

Trois critères :

1. *objet des connaissances* :
   - il existe des idées éternelles /
   - il n'existe que des substances corruptibles ;
2. *origine des connaissances* :
   - les choses peuvent être connues *a priori* (appartenance des concepts aux choses) /
   - on ne peut les connaître qu'*a posteriori* (extériorité des concepts aux choses) ;
3. *méthode des connaissances* :
   - les choses peuvent être connues en soi /
   - on n'a accès qu'à l'activité de notre subjectivité ;

(Critère subsidiaire : le monde est composé de choses / le monde est composé d'événements.)

Classification de Jules Vuillemin (1984)



Implication : idées éternelles $\rightarrow$ connaissance *a priori* et objective.

Classification d'après Vuillemin

Systèmes philosophiques	Langages de programmation
Réalisme : 1. objets éternels 2. inclusions de concepts	*Programmation logique* : 1. termes indestructibles 2. inclusions éternelles de concepts
Conceptualisme : 1. substances corruptibles 2. concepts immanents	*Orienté objet* : 1. objets créés et destructibles 2. fonctions internes («méthodes»)
Nominalisme des choses : 1. substances corruptibles 2. concepts extérieurs	*Impératif* : 1. variables créées et destructibles 2. fonctions externes
Nominalisme des événements : 1. pas d'objet 2. monde d'événéments	*Fonctionnel* : 1. pas d'assignation de variables 2. monde de fonctions

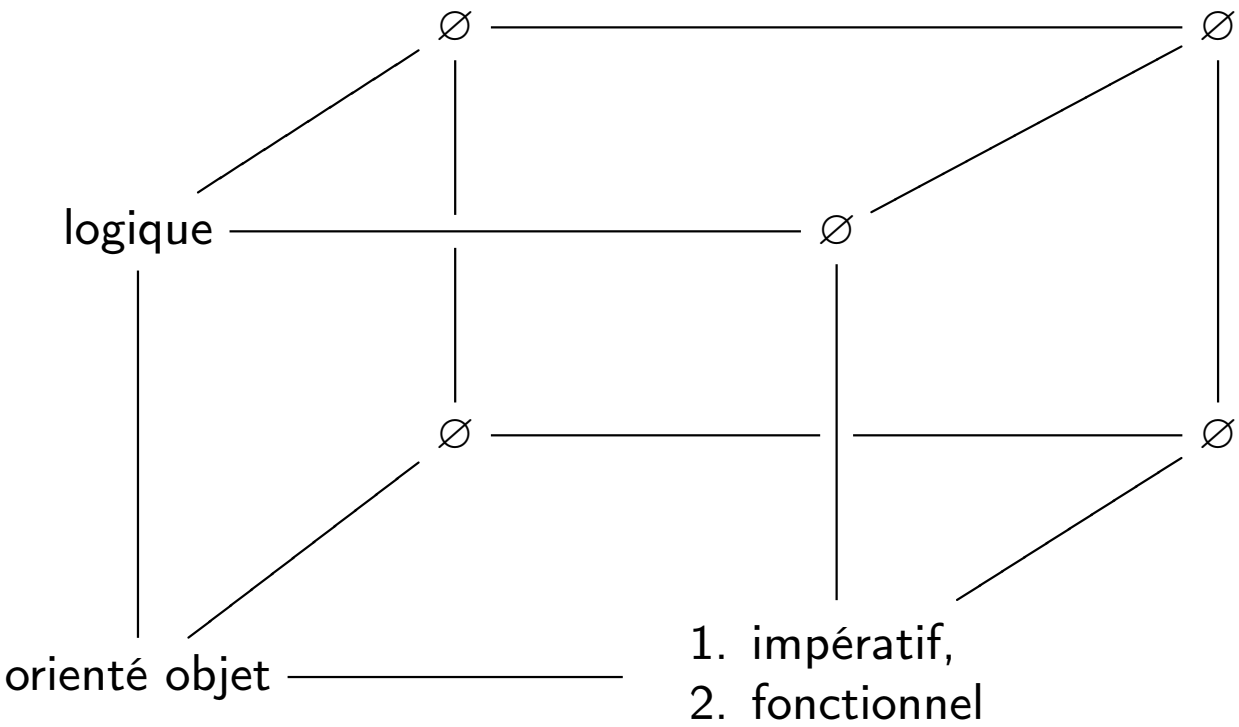
Classification d'après Vuillemin

1.

Fonction			Style
Éternelle (+) / temporelle (−)	Interne (+) externe (−)	avec (+) ou sans (−) objets	
+	+	+	Logique
−	+	+	Orienté objet
−	−	+	Impératif
−	−	−	Fonctionnel

2. Implications :
   1. fonction éternelle → interne ;
   2. fonction interne → avec objets.

Classification d'après Vuillemin



Bibliographie

1. John Backus, «Can programming be liberated from the von Neumann style?: a functional style and its algebra of programs», 1978.
2. Donald Knuth, *The Art of Computer Programming I*, 1997.
3. John von Neumann, «First Draft of a Report on the EDVAC», 1945.
4. Peter Rechenberg, «Programming Languages as Thought Models», 1990.
5. Raymond Turner et Amnon H. Eden, «Towards a Programming Language Ontology», 2007
6. Jules Vuillemin, *Nécessité ou contingence*, 1984.