

Gilles Dowek

UNDERGRADUATE TOPICS
in COMPUTER SCIENCE

Principles of Programming Languages

1. Imperative Core	1
1.1 Five Constructs	1
1.1.1 Assignment	1
1.1.2 Variable Declaration	3
1.1.3 Sequence	5
1.1.4 Test	6
1.1.5 Loop	6
1.2 Input and Output	7
1.2.1 Input	7
1.2.2 Output	7
1.3 The Semantics of the Imperative Core	8
1.3.1 The Concept of a State	8
1.3.2 Decomposition of the State	9
1.3.3 A Visual Representation of a State	10
1.3.4 The Value of Expressions	11
1.3.5 Execution of Statements	13
2. Functions	19
2.1 The Concept of Functions	19
2.1.1 Avoiding Repetition	19
2.1.2 Arguments	21
2.1.3 Return Values	22
2.1.4 The <i>return</i> Construct	23
2.1.5 Functions and Procedures	24
2.1.6 Global Variables	25
2.1.7 The Main Program	25

2.1.8	Global Variables Hidden by Local Variables	27
2.1.9	Overloading	28
2.2	The Semantics of Functions	29
2.2.1	The Value of Expressions	30
2.2.2	Execution of Statements	31
2.2.3	Order of Evaluation	34
2.2.4	CamL	34
2.2.5	C	36
2.3	Expressions as Statements	37
2.4	Passing Arguments by Value and Reference	37
2.4.1	Pascal	39
2.4.2	CamL	40
2.4.3	C	41
2.4.4	Java	45
3.	Recursion	47
3.1	Calling a Function from Inside the Body of that Function	47
3.2	Recursive Definitions	48
3.2.1	Recursive Definitions and Circular Definitions	48
3.2.2	Recursive Definitions and Definitions by Induction	49
3.2.3	Recursive Definitions and Infinite Programs	49
3.2.4	Recursive Definitions and Fixed Point Equations	51
3.3	CamL	53
3.4	C	54
3.5	Programming Without Assignment	55
4.	Records	59
4.1	Tuples with Named Fields	59
4.1.1	The Definition of a Record Type	60
4.1.2	Allocation of a Record	60
4.1.3	Accessing Fields	62
4.1.4	Assignment of Fields	62
4.1.5	Constructors	64
4.1.6	The Semantics of Records	65
4.2	Sharing	66
4.2.1	Sharing	66
4.2.2	Equality	68
4.2.3	Wrapper Types	68
4.3	CamL	73
4.3.1	Definition of a Record Type	73
4.3.2	Creating a Record	73
4.3.3	Accessing Fields	74

4.3.4	Assigning to Fields	74
4.4	C	76
4.4.1	Definition of a Record Type	76
4.4.2	Creating a Record	76
4.4.3	Accessing Fields	77
4.4.4	Assigning to Fields	77
4.5	Arrays	79
4.5.1	Array Types	79
4.5.2	Allocation of an Array	80
4.5.3	Accessing and Assigning to Fields	80
4.5.4	Arrays of Arrays	82
4.5.5	Arrays in Caml	83
4.5.6	Arrays in C	84
5.	Dynamic Data Types	85
5.1	Recursive Records	85
5.1.1	Lists	85
5.1.2	The null Value	86
5.1.3	An Example	86
5.1.4	Recursive Definitions and Fixed Point Equations	88
5.1.5	Infinite Values	89
5.2	Disjunctive Types	90
5.3	Dynamic Data Types and Computability	92
5.4	Caml	92
5.5	C	94
5.6	Garbage Collection	96
5.6.1	Inaccessible Cells	96
5.6.2	Programming without Garbage Collection	98
5.6.3	Global Methods of Memory Management	100
5.6.4	Garbage Collection and Functions	102
6.	Programming with Lists	103
6.1	Finite Sets and Functions of a Finite Domain	103
6.1.1	Membership	103
6.1.2	Association Lists	104
6.2	Concatenation: Modify or Copy	105
6.2.1	Modify	105
6.2.2	Copy	109
6.2.3	Using Recursion	111
6.2.4	Chemical Reactions and Mathematical Functions	111
6.3	List Inversion: an Extra Argument	112
6.4	Lists and Arrays	114

6.5	Stacks and Queues	114
6.5.1	Stacks	115
6.5.2	Queues	118
6.5.3	Priority Queues	119
7.	Exceptions	121
7.1	Exceptional Circumstances	121
7.2	Exceptions	122
7.3	Catching Exceptions	122
7.4	The Propagation of Exceptions	123
7.5	Error Messages	124
7.6	The Semantics of Exceptions	124
7.7	Caml	125
8.	Objects	127
8.1	Classes	127
8.1.1	Functions as Part of a Type	127
8.1.2	The Semantics of Classes	129
8.2	Dynamic Methods	129
8.3	Methods and Functional Fields	132
8.4	Static Fields	132
8.5	Static Classes	133
8.6	Inheritance	134
8.7	Caml	137
9.	Programming with Trees	139
9.1	Trees	139
9.2	Traversing a Tree	142
9.2.1	Depth First Traversal	143
9.2.2	Breadth First Traversal	145
9.3	Search Trees	146
9.3.1	Membership	146
9.3.2	Balanced Trees	149
9.3.3	Dictionaries	151
9.4	Priority Queues	152
9.4.1	Partially Ordered Trees	152
9.4.2	Partially Ordered Balanced Trees	153
	Index	157