Contents

Thanks **3**

Introduction **1**

1 Mozart and comprehensions **2**

1.1 Mozart2, a functional language 2

Syntactic sugars 2

Expressions and statements 2

Declarations 2

Variables 2

Functions and procedures 2

Records and tuples 2

Lists 2

If statements 2

Pattern matching 2

Recursion 3

Threads 3

Streams 3

Laziness 3

Cells 3

Classes and objects 3

Functors 3

1.2 List comprehensions 3

Principle 3

Terminology 3

1.3 Record comprehensions 4

Principle 4

Terminology 4

1.4 Complete syntax 4

List comprehensions 4

Record comprehensions 4

2 The compiler of Mozart2 **5**
2.1 Generic design of a compiler using a virtual machine . . . 5
Lexical analysis, the lexer . . . 5
Syntax analysis, the parser . . . 5
Semantic analysis . . . 5
Optimization . . . 5
Code generation . . . 5
Mozart2 . . . 5
2.2 The abstract syntax tree of Mozart2 . . . 5
Positions . . . 5
Basic types . . . 5
Equality . . . 6
Skip statement . . . 6
Local declarations . . . 6
Operations . . . 6
Procedure and function calls . . . 6
Procedure and function declarations . . . 6
If statements . . . 6
Threads . . . 6
Successive statements . . . 6
Records, tuples and lists . . . 6
Layers and range generators . . . 6
For loops . . . 6
Step Points . . . 7
2.3 New nodes of AST for comprehensions . . . 7
Bounded buffers . . . 7
Ranges generated by records . . . 7
Expressions . . . 7
Collects . . . 7
List comprehension levels . . . 7
List comprehensions . . . 7
Record comprehensions . . . 7
2.4 Syntactic sugars compilation in Mozart2 . . . 7
Macros . . . 7
The lexer, lexical analysis . . . 8
The parser, syntax analysis . . . 8
Checking the syntax . . . 8
Coordinates in case of failure . . . 8
The unnester, transforming syntactic sugars . . . 8

Keywords in Emacs 8

3 Functional transformations 9

3.1 Functional transformations 9

An inspiration, for loops examples 9

Multi level 9

Multi layer 9

Multi output 9

Output conditions 9

Integer generator 9

List generator 9

C style generator 9

From generator 9

Records as generators 10

Laziness 10

Record comprehensions 10

3.2 Implementation 10

Architecture 10

Applying the transformations 10

4 The tests 11

4.1 General tests 11

One level – one layer 11

One level – multi layer 11

Multi level – one layer 11

Multi level – multi layer 11

From 11

Records as generators 11

Multi output 11

Labeled output 11

Output conditions 11

Laziness 12

Miscellaneous 12

Collectors and body 12

Record comprehensions 12

4.2 Other languages 12

Erlang 12

Python 12

Haskell 12

4.3 Performance tests 12

Space analysis . . . 12
Time analysis . . . 12
4.4 Application examples . . . 13
Permutations . . . 13
Map . . . 13
Flatten . . . 13
Sort . . . 13
Factorial . . . 13

5 Concurrency using list comprehensions **14**
5.1 Laziness . . . 14
5.2 Producer − consumer . . . 14
5.3 Multi output and conditions . . . 14
5.4 Logic gates . . . 14
5.5 Bounded buffer . . . 14

Conclusion **15**
Possible evolutions . . . 15
Conclusion . . . 15

References **16**
Offline . . . 16
Online . . . 17