
CONTENTS

Acknowledgments	ix
1 Overview	1
1.1 Continuation-passing style	1
1.2 Advantages of CPS	4
1.3 What is ML?	6
1.4 Compiler organization	9
2 Continuation-passing style	11
2.1 The CPS datatype	11
2.2 Functions that escape	16
2.3 Scope rules	17
2.4 Closure conversion	18
2.5 Spilling	21
3 Semantics of the CPS	23
4 ML-specific optimizations	37
4.1 Data representation	37
4.2 Pattern matching	43
4.3 Equality	45
4.4 Unboxed updates	46
4.5 The mini-ML sublanguage	46
4.6 Exception declarations	49
4.7 The lambda language	50
4.8 The module system	52
5 Conversion into CPS	55
5.1 Variables and constants	55
5.2 Records and selection	56
5.3 Primitive arithmetic operators	57
5.4 Function calls	59
5.5 Mutually recursive functions	60
5.6 Data constructors	60
5.7 Case statements	61

vi	CONTENTS
5.8	Exception handling 63
5.9	Call with current continuation 64
6	Optimization of the CPS 67
6.1	Constant folding and β -contraction 68
6.2	Eta reduction and uncurrying 76
6.3	Cascading optimizations 78
6.4	Implementation 80
7	Beta expansion 83
7.1	When to do in-line expansion 87
7.2	Estimating the savings 89
7.3	Runaway expansion 92
8	Hoisting 93
8.1	Merging FIX definitions 93
8.2	Rules for hoisting 95
8.3	Hoisting optimizations 96
9	Common subexpressions 99
10	Closure conversion 103
10.1	A simple example 104
10.2	A bigger example 106
10.3	Closure-passing style 109
10.4	The closure-conversion algorithm 109
10.5	Closure representation 112
10.6	Callee-save registers 114
10.7	Callee-save continuation closures 119
10.8	Stack allocation of closures 122
10.9	Lifting function definitions to top level 124
11	Register spilling 125
11.1	Rearranging the expression 128
11.2	The spilling algorithm 128
12	Space complexity 133
12.1	Axioms for analyzing space 136
12.2	Preserving space complexity 137
12.3	Closure representations 142
12.4	When to initiate garbage collection 144
13	The abstract machine 147
13.1	Compilation units 147
13.2	Interface with the garbage collector 148
13.3	Position-independent code 150

<i>CONTENTS</i>	<i>vii</i>
13.4 Special-purpose registers	151
13.5 Pseudo-operations	154
13.6 Instructions of the continuation machine	155
13.7 Register assignment	158
13.8 Branch prediction	160
13.9 Generation of abstract-machine instructions	161
13.10 Integer arithmetic	161
13.11 Unboxed floating-point values	162
14 Machine-code generation	165
14.1 Translation for the VAX	165
14.1.1 Span-dependent instructions	167
14.2 Translation for the MC68020	168
14.3 Translation for the MIPS and SPARC	169
14.3.1 PC-relative addressing	170
14.3.2 Instruction scheduling	170
14.3.3 Anti-aliasing	171
14.3.4 Alternating temporaries	173
14.4 An example	174
15 Performance evaluation	179
15.1 Hardware	181
15.2 Measurements of individual optimizations	183
15.3 Tuning the parameters	187
15.4 More about caches	187
15.5 Compile time	198
15.6 Comparison with other compilers	200
15.7 Conclusions	201
16 The runtime system	205
16.1 Efficiency of garbage collection	205
16.2 Breadth-first copying	206
16.3 Generational garbage collection	207
16.4 Runtime data formats	210
16.5 Big bags of pages	211
16.6 Asynchronous interrupts	212
17 Parallel programming	215
17.1 Coroutines and semaphores	216
17.2 Better programming models	219
17.3 Multiple processors	220
17.4 Multiprocessor garbage collection	221

viii	CONTENTS
18 Future directions	223
18.1 Control dependencies	223
18.2 Type information	225
18.3 Loop optimizations	225
18.4 Garbage collection	227
18.5 Static single-assignment form	228
18.6 State threading	228
A Introduction to ML	229
A.1 Expressions	231
A.2 Patterns	233
A.3 Declarations	235
A.4 Some examples	236
B Semantics of the CPS	239
C Obtaining Standard ML of New Jersey	245
C.1 By Internet FTP	246
C.2 Without using the Internet	247
D Readings	249
Bibliography	251
Index	258