Contents

Preface **v**

I A++ – The Language **1**

1 Educational Programming Languages **3**
1.1 Introduction 3
1.2 Pascal 3
1.3 Scheme 3
1.4 Logo 4
1.5 A++ 4

2 Introduction to A++ **5**
2.1 Purpose of A++ and Origin 5
2.1.1 Purpose 5
2.1.2 Origin 5
2.1.3 ARS — *Generalization of the Lambda-Calculus* 5
2.1.4 Name of the language 6
2.2 Motivations for the development of A++ 6
2.2.1 To support an alternate method of teaching the principles of programming 6
2.2.2 To provide a learning tool for exploring the fundamentals of programming 7
2.2.3 To support a method teaching powerful programming patterns applicable to most languages 7
2.2.4 To open a new view of programming for many programmers: 7
2.3 Features of A++ 7
2.3.1 Programming paradigms supported: 7
2.3.2 Constitutive Principles of A++ 7
2.3.3 Closure 8
2.3.4 Basic abstractions derived from ARS 9
2.3.5 Development of Applications with A++ 9
2.4 Internal Architecture of A++ 10
2.4.1 Overview 10
2.4.2 Internal Structure Overview (commented version) 11
2.4.3 Syntax of ARS (A++) 11
2.4.4 Definition of A++ in the form of a Tree Diagram 13
2.4.5 Commenting the definition: 13
2.4.6 Examples of A++ - Syntax 14

3 Linking Logic with the Physical World **15**
3.1 Introduction 15
3.2 Internal Structure of A++ Including Minimal Set of Primitives 16
3.2.1 Overview 16

3.2.2 Internal Structure Overview (commented version) . . . 17
3.2.3 Syntax of A++ with Minimal Set of Primitives . . . 19
3.3 Examples using pre-defined primitives of A++ . . . 21
3.3.1 Pre-defined primitive abstractions in A++ . . . 21
3.3.2 A++ including pre-defined primitives . . . 22
3.3.3 A++ Overview . . . 23

4 Program Simplicity plus Data Structure Simplicity 25
4.1 Introducing JSON . . . 25
4.1.1 Practical data types or data constructs . . . 25
4.1.2 Syntax of JSON . . . 26
4.2 Integrating JSON into A++ . . . 27
4.2.1 Structure of A++ including JSON (Overview) . . . 27
4.2.2 Structure of A++ including JSON (commented version) . . . 29
4.2.3 Syntax of A++ merged with JSON Architecture . . . 32

5 General Programming Patterns and A++ 37
5.1 Closure Pattern . . . 37
5.2 CLAM Pattern . . . 38
5.3 List Pattern . . . 38
5.4 Dictionary Pattern . . . 39
5.5 Set Pattern . . . 40
5.6 Recursion Pattern . . . 40
5.7 Higher Order Function Pattern . . . 41
5.8 Message Passing Pattern . . . 41
5.8.1 Classes of objects . . . 41
5.8.2 Instance of a class . . . 42
5.8.3 Constructors . . . 42
5.8.4 Creating instances of a class . . . 43
5.8.5 Sending messages . . . 44
5.8.6 Executing methods . . . 45
5.8.7 Essential features of object oriented programming . . . 46
5.8.8 Relation between classes . . . 47
5.9 Meta Object Protocol Pattern . . . 47

6 Discovering the Power of A++ 49
6.1 Basic Abstractions . . . 49
6.1.1 The IF- Abstraction . . . 49
6.1.2 Extended Logical Abstractions . . . 50
6.1.3 Application of extended logical abstractions . . . 51
6.2 Numeric Abstractions . . . 51
6.2.1 Natural Numbers . . . 51
6.2.2 Arithmetic Operations . . . 53
6.2.3 Application of numeric abstractions . . . 54
6.3 Collections of Data . . . 54
6.3.1 Basic abstractions for pairs . . . 55
6.3.2 Basic utility abstractions for lists . . . 56
6.4 Extended Numerical Abstractions . . . 57
6.4.1 Abstraction ‘zeropair’ . . . 57
6.4.2 Decrementing a number: ‘pred’ . . . 57
6.4.3 Subtracting a number: ‘sub’ . . . 58
6.5 Relational Abstractions . . . 58
6.5.1 Comparing two numbers: ‘equaln’ . . . 58

6.5.2 Comparing two numbers: 'gtp' . . . 58
6.5.3 Comparing two numbers: 'ltp' . . . 58
6.5.4 Comparing two numbers: 'gep' . . . 59
6.6 Examples for recursion . . . 59
6.6.1 Calculating the faculty of a number . . . 59
6.6.2 Calculating the sum of elements of a list . . . 59
6.6.3 Insertion Sort . . . 59
6.7 Higher Order Functions . . . 60
6.7.1 Creating a new function by composition: 'compose' . . . 60
6.7.2 Changing the arity of a function:'curry' . . . 60
6.7.3 Converting all elements in a list: 'map' . . . 60
6.7.4 Converting the 'map' function: 'mapc' . . . 61
6.7.5 Selecting elements from a given list: 'filter' . . . 62
6.7.6 Searching for an element in a given list: 'locate' . . . 62
6.7.7 Iterating through all elements of a list: 'for-each' . . . 62
6.8 Set Operations . . . 63
6.8.1 Checking for a member in a set: 'memberp' . . . 63
6.8.2 Adding an element to a set: 'addelt' . . . 63
6.8.3 Combining two sets: 'union' . . . 64
6.9 Associative Lists in A++ . . . 64
6.9.1 Abstractions for associative lists . . . 65
6.9.2 Application of associative lists . . . 65
6.10 Miscellaneous abstractions . . . 66
6.10.1 Testing the basic abstractions . . . 66
6.11 Object Oriented Programming in A++ . . . 68
6.11.1 First example of object oriented programming in A++ . . . 68
6.11.2 Second example of object oriented programming in A++ . . . 69
6.11.3 Third example of object oriented programming in A++ . . . 75
6.12 Imperative Programming in A++ . . . 83

7 Computer Resources for A++ 85
7.1 Support Functions . . . 85
7.1.1 Abstraction for displaying a number . . . 85
7.1.2 Abstraction for displaying a boolean value . . . 85
7.1.3 Abstraction for displaying lists . . . 85
7.2 A++ Interpreters . . . 85
7.2.1 A++ Interpreter written in Perl . . . 86
7.2.2 A++ Interpreter written in C . . . 90
7.3 Initializing the A++ Interpreter . . . 92
7.3.1 Initializing the A++ Interpreter part 1 . . . 92
7.3.2 Initializing the A++ Interpreter part 2 . . . 92
7.3.3 Initializing the A++ Interpreter part 3 . . . 92
7.3.4 Initializing the A++ Interpreter part 4 . . . 93
7.3.5 Initializing the A++ Interpreter part 5 . . . 93
7.3.6 Initializing the A++ Interpreter part 6 . . . 94
7.3.7 Initializing the A++ Interpreter part 7 . . . 94
7.4 WWW Links . . . 95

8 Extending A++ 97
8.1 ARS++ . . . 97
8.2 ARSAPI . . . 98

II A++ – The Implementation 99

1 General Introduction 101

2 ARS and Perl 103
2.1 Syntax of Perl . . . 103
2.2 Syntax of ARS . . . 104
2.2.1 Examples of syntax of abstraction . . . 104
2.2.2 Examples of syntax of reference . . . 104
2.2.3 Examples of syntax of synthesis . . . 104
2.3 Basic Abstractions . . . 104
2.3.1 The IF- Abstraction . . . 104
2.3.2 The boolean values 'true' and 'false' . . . 105
2.3.3 Application of basic logical abstractions . . . 105
2.3.4 Extended Logical Abstractions . . . 105
2.3.5 Application of extended logical abstractions . . . 105
2.4 Numeric Abstractions . . . 106
2.5 Arithmetic Operations . . . 106
2.6 Application of numeric abstractions . . . 106
2.7 Collections of Data . . . 106
2.7.1 Basic abstractions for pairs . . . 107
2.7.2 Application of abstractions for pairs . . . 107
2.8 Basic utility abstractions for lists . . . 108
2.8.1 Checking for an empty list: 'nullp' . . . 108
2.8.2 Displaying the contents of a list: 'ldisp' . . . 108
2.8.3 Determining the length of a list: 'length' . . . 109
2.8.4 Removing an element from a list: 'remove' . . . 109
2.8.5 Retrieving the n-th element from a list: 'nth' . . . 109
2.9 Relational Abstractions . . . 110
2.10 Examples for recursion . . . 110
2.10.1 Calculating the faculty of a number . . . 110
2.10.2 Calculating the sum of elements of a list . . . 110
2.11 Higher Order Functions . . . 111
2.11.1 Creating a new function by composition: 'compose' . . . 111
2.11.2 Changing the arity of a function:'curry' . . . 111
2.11.3 Converting all elements in a list: 'map1' . . . 111
2.11.4 Converting the 'map1' function: 'mapc' . . . 112
2.11.5 Selecting elements from a given list: 'filter' . . . 112
2.11.6 Iterating through all elements of a list: 'for-each' . . . 113
2.12 Set Operations . . . 113
2.12.1 Checking for a member in a set: 'memberp' . . . 113
2.12.2 Adding an element to a set: 'addelt' . . . 114
2.12.3 Combining two sets: 'union' . . . 114
2.13 Associative lists . . . 115
2.13.1 Native lists or arrays . . . 115
2.13.2 Abstractions for associative lists . . . 117
2.13.3 Application of associative lists . . . 117
2.14 Object Oriented Programming in Perl . . . 119
2.14.1 Object Oriented Programming as Introduced in ARS . . . 119
2.14.2 Modules support object-oriented programming in Perl . . . 120
2.15 Regular expressions . . . 122

3 A++ Interpreter in Perl **123**
3.1 Introduction . . . 123
3.2 A++ Interpreter . . . 125
3.2.1 Main Program: Command Line Mode . . . 125
3.2.2 Main Program: WEB-Based Application . . . 127
3.2.3 Expression Definition Module (EXP) . . . 128
3.2.4 List Module (PAIR) . . . 129
3.2.5 Value Module (VALUE) . . . 131
3.2.6 Name Management Module (NAME) . . . 132
3.2.7 Environment Module (ENV) . . . 133
3.2.8 Parser Module (ARSP) . . . 137
3.2.9 Expression Evaluation Module (EVAL) . . . 141

4 ARS and C **149**
4.1 Mechanisms for Abstraction . . . 149
4.1.1 Give something a name . . . 149
4.1.2 Abstract Data Types (ADT) . . . 150
4.1.3 Defining Functions . . . 150
4.2 Operations within functions . . . 150
4.2.1 Basic operations . . . 150
4.2.2 Definition of variables . . . 150
4.2.3 Assignment of values . . . 151
4.2.4 Defining functions . . . 151
4.2.5 Invoking functions . . . 152
4.2.6 Testing values . . . 152
4.2.7 Repeating operations . . . 153
4.2.8 Arithmetic operations . . . 154
4.2.9 Relational operations . . . 154
4.2.10 Boolean operations . . . 155
4.2.11 Library functions . . . 155
4.2.12 Testing the basic operations . . . 155
4.3 Simple and complex expressions . . . 156
4.3.1 Atom . . . 156
4.3.2 Complex expression . . . 156
4.4 Modularization . . . 156
4.5 Variables and pointers . . . 157
4.6 Structures in C . . . 159
4.6.1 typedef . . . 159
4.6.2 enum . . . 159
4.6.3 struct . . . 159
4.6.4 union . . . 160
4.7 Special control operations: setjmp/longjmp . . . 160

5 A++ Interpreter in C **161**
5.1 Introduction . . . 161
5.2 ADT's used by the the interpreter . . . 161
5.2.1 NAME . . . 163
5.2.2 PRIM . . . 163
5.2.3 SPEC . . . 164
5.2.4 VTYPE . . . 164
5.2.5 EXPTYPE . . . 165
5.2.6 ABS . . . 165

5.2.7 SYN 165
5.2.8 EXP 165
5.2.9 ELIST 166
5.2.10 NLIST 166
5.2.11 VLIST 166
5.2.12 ENV 166
5.2.13 LFUN 167
5.2.14 CLAM 167
5.2.15 ACL 168
5.2.16 THUNK 168
5.2.17 VALUE 168
5.3 Functions used by the interpreter 169
5.3.1 Constructors 169
5.3.2 Name management 171
5.3.3 Environments 171
5.3.4 Evaluation of expressions 172
5.3.5 User input interface 175
5.3.6 Reading of A++ code by the parser 175
5.3.7 Main program of the A++ Interpreter 176
5.4 Source code of the A++ Interpreter 177
5.4.1 Main ARS-Module Header File: Datatypes and Function-Prototypes 177
5.4.2 Main ARS-Module Implementation File: arsc.c 183
5.4.3 Environment Modul ENV: app_env.c 192
5.4.4 A++-Interface for Primitive Functions: app_prim.c 196
5.4.5 Primitive Functions related to data type boolean: app_bool.c 202
5.4.6 Primitive Functions related to data type string: app_string.c 203
5.4.7 Primitive Functions related to data type char: app_char.c 205
5.4.8 Primitive Functions related to data type integer and double: app_number.c . 206
5.4.9 A++/JSON-Interface: app_jso.c 209
5.4.10 ARS-Parser Module: arsparser.c 213
5.4.11 Primitive functions related to I/O ports: Modul: app_port.c 219
5.4.12 I/O Modul: app_read.c 221
5.4.13 I/O Modul: app_print.c 222
5.4.14 Generic Value Modul: app_val.c 226
5.4.15 Error Modul: app_error.c 227
5.4.16 Main Program 228
5.4.17 Script for building the interpreter 229

6 ARS based programming in C 231
6.1 The ‘clam’ as a key for ARS based programming in C 231
6.2 Datatypes 231
6.2.1 Datatype ‘NAME’ 231
6.2.2 nameTable 232
6.2.3 Abstract Datatype ‘VALUE’ 232
6.2.4 Datatype ‘INTV’ 233
6.2.5 Datatype ‘SYMV’ 234
6.2.6 Datatype ‘STRV’ 234
6.2.7 Datatype ‘PAIRV’ 234
6.2.8 Datatype ‘NLIST’ 235
6.2.9 Datatype ‘VLIST’ 236
6.2.10 Datatype ‘ENV’ 236
6.2.11 Datatype ‘CLAM’ 237

6.3 Mapping ARS to C . . . 238
6.3.1 Abstraction . . . 238
6.3.2 Reference . . . 238
6.3.3 Synthesis . . . 238
6.4 Simple example . . . 238
6.5 Basic demo of ARSAPI for C . . . 240
6.5.1 Main program . . . 240
6.5.2 Testing object oriented programming in C . . . 240
6.5.3 Testing functional programming . . . 240
6.6 Advanced demo of ARSAPI for C: menu system 'genfui' . . . 240
6.6.1 Source code of 'ARSAPI for C' demos . . . 241
6.6.2 Header files for the demo . . . 241
6.6.3 Library for ARS based programming in C . . . 241
6.6.4 Basic demo main program . . . 241

Appendices **251**

A Detailed Discussion of Addition and Multiplication **251**
A.1 Addition of the numbers 'two' and 'three' . . . 251
A.2 Multiplication of the numbers 'two' and 'three' . . . 253

B The Lambda Calculus **257**
B.1 Introduction . . . 257
B.1.1 Origin . . . 257
B.1.2 Definition . . . 257
B.1.3 Literature . . . 257
B.2 Syntax of Lambda Expressions . . . 257
B.3 Basic Rules for Lambda Conversions . . . 258
B.3.1 Notation used in Conversion Rules . . . 258
B.3.2 Alpha Conversion . . . 258
B.3.3 Beta Conversion . . . 259
B.3.4 Eta Conversion . . . 260
B.3.5 Rules of Associativity . . . 261
B.3.6 Y-Combinator . . . 262

C Testing the Y-Combinator in A++ **265**
C.1 A++ Source code of Y-Combinator test program . . . 265
C.2 Running the Y-Combinator test program . . . 265
C.3 Comments on the Y-Combinator test program . . . 266

D Background of the Author **267**