

The Automatic Design of Batch Processing Systems

by

Barry Dwyer, M.A., D.A.E., Grad.Dip.

*A thesis submitted for the degree of
Doctor of Philosophy
in the Department of Computer Science
University of Adelaide*

February 1999

Abstract

Batch processing is a means of improving the efficiency of transaction processing systems. Despite the maturity of this field, there is no rigorous theory that can assist in the design of batch systems. This thesis proposes such a theory, and shows that it is practical to use it to automate system design. This has important consequences; the main impediment to the wider use of batch systems is the high cost of their development and maintenance. The theory is developed twice: informally, in a way that can be used by a systems analyst, and formally, as a result of which a computer program has been developed to prove the feasibility of automated design.

Two important concepts are identified, which can aid in the decomposition of any system: ‘separability’, and ‘independence’. Separability is the property that allows processes to be joined together by pipelines or similar topologies. Independence is the property that allows elements of a large set to be accessed and updated independently of one another. Traditional batch processing technology exploits independence when it uses sequential access in preference to random access. It is shown how the same property allows parallel access, resulting in speed gains limited only by the number of processors. This is a useful development that should assist in the design of very high throughput transaction processing systems.

Systems are specified procedurally by describing an ideal system, which generates output and updates its internal state immediately following each input event. The derived systems have the same external behaviour as the ideal system except that their outputs and internal states lag those of the ideal system arbitrarily. Indeed, their state variables may have different delays, and the systems as whole may never be in consistent state.

A ‘state dependency graph’ is derived from a static analysis of a specification. The reduced graph of its strongly-connected components defines a canonical process network from which all possible implementations of the system can be derived by composition. From these it is possible to choose the one that minimises any imposed cost function. Although, in general, choosing the optimum design proves to be an NP-complete problem, it is shown that heuristics can find it quickly in practical cases.

Declaration

This thesis contains no material that has been accepted for the award of any other degree or diploma in any university or other tertiary institution. To the best of my knowledge and belief, it contains no material previously published or written by any other person, except where due reference is made in the text.

I give consent for this copy of my thesis, when deposited in the University Library, to be available for loan and photocopying.

Barry Dwyer
February 1999

Acknowledgments

Although I have to admit that I have learned a lot in the process, completing this thesis has taken me an excruciatingly long time. I am therefore grateful for the faith and patience of my supervisor, Prof. C.J. Barter. I am even more grateful for the support of my wife, Linda, who has often had to forego the pleasure of my company while I worked on it. In addition, I would like to thank those of my friends with whom I have discussed my work. Even if they thought they weren't helping me, the act of explaining my ideas helped me clarify many difficult issues. Finally, I would like to thank the reviewers of my thesis for suggesting how it could be improved. I have done my best to follow their suggestions faithfully.

1. INTRODUCTION	1
1.1 THE AIM OF THE THESIS	1
1.2 TRANSACTION PROCESSING SYSTEMS	3
1.3 BATCH SYSTEMS.....	4
1.4 THE MACROTOPIAN REFERENCE LIBRARY	5
1.5 MODELLING THE LIBRARY.....	8
1.6 SEQUENTIAL ACCESS.....	12
1.7 SCOPE OF THE THESIS.....	12
1.8 ORGANISATION OF THE THESIS	14
2. SYSTEM DESCRIPTION	17
2.1 SYSTEM MODELS	18
2.2 PROCESS GRAPHS	21
2.3 DELAYED AND LOCAL PROCEDURE CALLS.....	22
2.4 DATA STRUCTURES.....	24
2.5 A SPECIFICATION EXAMPLE.....	26
2.6 THE SPECIFICATION LANGUAGE.....	33
2.6.1 <i>External Interface</i>	33
2.6.2 <i>Variables</i>	35
2.6.3 <i>Procedures</i>	36
2.6.4 <i>Statements</i>	36
1.7 REWRITING RULES.....	40
3. IMPLEMENTATION	45
3.1 MODELLING INDEPENDENT UPDATING	46
3.2 RANDOM ACCESS IMPLEMENTATION	48
3.3 SEQUENTIAL ACCESS IMPLEMENTATION.....	52
3.4 ANALYSIS OF SEQUENTIAL AND RANDOM ACCESS	59
3.5 A PARALLEL UPDATE ALGORITHM	62
3.6 ANALYSIS OF THE PARALLEL UPDATE ALGORITHM	65
3.7 COMPOSITE KEYS.....	70
3.8 LIMITATIONS.....	71
4. SEPARABILITY	75
4.1 REAL-TIME EQUIVALENCE.....	75
4.2 EVENT TIMING.....	77
4.3 TIMING OF SEPARABLE PROCESSES	79
4.4 DEPENDENCE	82
4.5 A WORKING DEFINITION OF DEPENDENCE.....	85
4.6 STATE DEPENDENCE GRAPHS	87
4.7 THE CANONICAL MINIMAL PROCESS GRAPH	91
1.8 FINDING THE CANONICAL PROCESS GRAPH.....	97
1.9 A WORKED EXAMPLE	101

5. INDEPENDENCE	109
5.1 THE TREATMENT OF NESTED LOOPS	111
5.2 UPDATING MULTIPLE ELEMENTS.....	113
5.3 SHOWING INDEPENDENCE ON STATE DEPENDENCE GRAPHS	117
5.4 COMPATIBILITY AND CONFLICT	120
5.5 SHOWING COMPATIBILITY INFORMATION IN GRAPHS	124
5.6 RECURSIVE STRUCTURES	127
6. REQUIREMENTS ANALYSIS	131
6.1 AN ORDER-PROCESSING SYSTEM.....	131
6.2 PROCESS COMPOSITION	134
6.3 ADDING OTHER KINDS OF EVENT	136
6.4 AVOIDING DATA-DEPENDENCE CYCLES.....	138
6.5 SENSITIVITY OF DESIGN TO SPECIFICATION	141
6.6 SENSITIVITY TO DATA REPRESENTATION	143
6.7 OPTIMISTIC FAIL-SAFE SYSTEMS	147
6.8 ERROR DETECTION AND CORRECTION	151
6.9 THE NEED FOR AUTOMATED DESIGN.....	153
7. SYSTEM GENERATION	157
7.1 EVENT DECOMPOSITION	157
7.2 TRANSFORMATION TO INDEPENDENT ACCESS	166
7.3 LOOP STRUCTURES	168
7.4 FREQUENCY.....	172
7.5 INCOMPATIBLE LOOPS	174
8. USE-DEFINITION ANALYSIS	177
8.1 INTRODUCTION	177
8.2 THE TREATMENT OF INDEXED VARIABLES	178
8.3 STATE VARIABLES AND LOCAL VARIABLES.....	183
8.4 AN ANALYSIS ALGORITHM.....	185
8.5 THE ANALYSIS OF COMPOUND STATEMENTS	186
8.6 THE ANALYSIS OF EXPRESSIONS	186
8.7 THE ANALYSIS OF ASSIGNMENTS	186
8.8 THE ANALYSIS OF 'IF' STATEMENTS	187
8.9 THE ANALYSIS OF PROCEDURE CALLS.....	191
8.10 THE ANALYSIS OF RETURN STATEMENTS	192
8.11 THE ANALYSIS OF LOOPS.....	192
8.11.1 <i>While Loops</i>	193
8.11.2 <i>All Loops</i>	197
8.11.3 <i>For Loops</i>	199
9. OPTIMISATION	201
9.1 THE COMPOSITION PROBLEM.....	201

Contents

9.2	THE LATTICE OF PROCESS GRAPHS	201
1.3	TWO EXAMPLE PROBLEMS	205
1.4	THE COST FUNCTION.....	206
1.5	A GREEDY HEURISTIC METHOD.....	209
1.6	THE COMPLEXITY OF THE OPTIMISATION PROBLEM.....	212
1.7	BRANCH-AND-BOUND SEARCH.....	214
1.8	HYBRID SEARCH	220
1.9	PROCESS COMPATIBILITY	222
10.	THE DESIGNER CASE TOOL	227
10.1	FORMAL SYNTAX.....	228
1.2	THE PARSER	229
1.2.1	<i>Variables</i>	232
1.2.2	<i>Statements</i>	232
1.3	THE DEPENDENCE ANALYSER	233
1.4	THE CANONISER.....	237
1.5	THE OPTIMISER.....	239
1.5.1	<i>The Degree of Independence</i>	240
1.5.2	<i>Preserving Structure</i>	243
1.5.3	<i>Optimiser Heuristics</i>	245
1.6	THE GENERATOR.....	248
1.7	CONCLUSIONS	249
11.	DISCUSSION	253
11.1	COMPARISON WITH OTHER METHODOLOGIES.....	253
11.1.1	<i>Entity-Relationship Modelling</i>	253
11.1.2	<i>Structured Analysis and System Specification</i>	256
11.1.3	<i>Jackson System Development</i>	259
11.1.4	<i>Advantages of the Canonical Decomposition Method</i>	263
11.2	THE SPECIFICATION PROBLEM.....	264
11.2.1	<i>Data Flow as a Starting Point</i>	264
11.2.2	<i>Non-Procedural Specification</i>	267
11.2.3	<i>Eliminating States</i>	268
11.3	EXTENSIONS OF THE METHOD	269
11.3.1	<i>Real-time Systems</i>	269
11.3.2	<i>Avoiding Deadlock in Databases</i>	272
11.4	CONTRIBUTIONS OF THE THESIS	274
11.4.1	<i>Generalisation of Sequential Update Algorithm</i>	275
11.4.2	<i>Parallel Batch Processing</i>	275
11.4.3	<i>Real-Time Equivalence</i>	275
11.4.4	<i>Separability</i>	275
11.4.5	<i>Independence</i>	275
11.4.6	<i>Extensions to Use-Definition Analysis</i>	275
11.4.7	<i>The State Dependence Graph</i>	276

11.4.8	<i>System Design as a Composition Problem</i>	276
11.4.9	<i>System Optimisation as a Lattice Problem</i>	276
11.4.10	<i>NP-Completeness of Process Optimisation</i>	276
11.4.11	<i>Optimisation Heuristics</i>	276
11.4.12	<i>The Designer Program</i>	276
11.4.13	<i>A Design Methodology</i>	277
11.5	SOME FRANK REMARKS.....	277
12.	REFERENCES	281
13.	APPENDIX: THE DESIGNER PROGRAM	290
13.1	THE PARSER	290
13.2	SHARED PREDICATES USED BY OTHER MODULES.....	297
13.3	THE DEPENDENCE ANALYSER	305
13.4	THE CANONISER.....	312
13.5	THE OPTIMISER.....	313
13.6	THE GENERATOR.....	320
INDEX		325

List of Figures

FIGURE 1.1.1: A PROCESS GRAPH.....	1
FIGURE 1.4.1: A DATA-FLOW DIAGRAM.....	6
FIGURE 1.4.2: A SECOND DATA-FLOW DIAGRAM.....	7
FIGURE 1.5.1: A PROCESS PIPELINE.....	9
FIGURE 1.5.2: EXPLOITING PARALLEL CONCURRENCY	10
FIGURE 2.2.1: A PROCESS GRAPH.....	21
FIGURE 3.1.1: A MODEL OF INDEPENDENT ACCESS	46
FIGURE 3.4.1: THE BREAK EVEN BETWEEN RANDOM AND SEQUENTIAL UPDATING.....	61
FIGURE 3.5.1: A MODEL OF THE IMPLEMENTATION.....	64
FIGURE 3.6.1: PARALLEL UPDATING COMPARED WITH THE BENCHMARK.....	66
FIGURE 3.6.2: PARALLEL QUERIES COMPARED WITH THE BENCHMARK.....	67
FIGURE 4.1.1: AN IDEAL SYSTEM TIME LINE	75
FIGURE 4.1.2: A MORE REALISTIC SYSTEM TIME LINE	75
FIGURE 4.1.3: TWO SYSTEM COMPONENTS	76
FIGURE 4.3.1: DELAYED PROCEDURE CALL.....	80
FIGURE 4.3.2: A CYCLE OF QUEUES.....	80
FIGURE 4.3.3: A PROCESS GRAPH.....	81
FIGURE 4.3.4: A PIPE-LINE.....	81
FIGURE 4.6.1: SDG FOR THE BORROW EVENT.....	88
FIGURE 4.6.2: SDG FOR THE REVISED LIBRARY SYSTEM	89
FIGURE 4.7.1: STRONG COMPONENTS OF A GRAPH	91
FIGURE 4.7.2: THE COMPONENT GRAPH OF FIGURE 4.7.1	92
FIGURE 4.7.3: A PROCESS GRAPH DERIVED FROM FIGURE 4.7.2.....	92
FIGURE 4.7.4: AN INVALID COMPOSITION BASED ON FIGURE 4.7.2.....	93
FIGURE 4.7.5: THE TRANSITIVE ROOT OF FIGURE 4.7.2.	93
FIGURE 4.7.6: A TOPOLOGICAL SORT OF FIGURE 4.7.5.....	94
FIGURE 4.8.1: STRONG COMPONENTS OF A GRAPH	99
FIGURE 4.9.1: SDG FOR THE STUDENT RECORD SYSTEM	102
FIGURE 4.9.2: CANONICAL PROCESS GRAPH FOR THE STUDENT RECORD SYSTEM	103
FIGURE 4.9.3: PROCESS PIPELINE FOR THE STUDENT RECORD SYSTEM.....	103
FIGURE 4.9.4: PARALLEL PROCESSES FOR THE STUDENT RECORD SYSTEM.....	103
FIGURE 5.3.1: MODELLING PARALLELISM.....	118
FIGURE 5.3.2: THE ‘TRANSFER’ SDG.....	119
FIGURE 5.3.3: THE ‘SAFE TRANSFER’ SDG.....	119
FIGURE 5.3.4: THE ‘CAREFUL TRANSFER’ SDG.....	119
FIGURE 5.3.5: ‘CAREFUL TRANSFER’ PLUS BUDGET CHECKING	120
FIGURE 5.5.1: AUDITING CLASS SIZES	125
FIGURE 5.5.2: THE ‘TRANSFER’ SDG.....	126
FIGURE 5.5.3: THE ‘SAFE TRANSFER’ SDG.....	126
FIGURE 5.5.4: THE ‘CAREFUL TRANSFER’ SDG.....	126
FIGURE 5.5.5: ‘CAREFUL TRANSFER’ AND BUDGET CHECKING	127
FIGURE 5.6.1: STRUCTURE OF A PAIR OF SPECTACLES	128
FIGURE 5.6.2: ONE ITERATION OF THE ‘REQUIREMENTS’ PROCEDURE.....	130
FIGURE 6.1.1: THE SDG FOR AN ORDER.....	132

FIGURE 6.1.2: THE SIMPLIFIED SDG	134
FIGURE 6.2.1: THE BEST GROUPING OF ATTRIBUTES.....	135
FIGURE 6.2.2: THE COMPOSITE PROCESS GRAPH.....	135
FIGURE 6.3.1: A DEPENDENCE DUE TO ‘CLOSE’ EVENTS	137
FIGURE 6.3.2: A CYCLE DUE TO ‘CLOSE’ EVENTS	137
FIGURE 6.3.3: SEPARABLE PROCESSES WITH ‘CLOSE’ EVENTS.....	138
FIGURE 6.4.1: SDG WITH A ‘CLOSED’ ATTRIBUTE	141
FIGURE 6.5.1: ORDERS WITH A CREDIT LIMIT	142
FIGURE 6.6.1: ORDERS WITH COMMITMENTS.....	144
FIGURE 6.6.2: ORDERS WITH CREDIT USED.....	144
FIGURE 6.6.3: COMPOSITE PROCESSES FOR ORDERS WITH CREDIT USED.....	145
FIGURE 6.6.4: PROCESS GRAPH FOR ORDERS WITH CREDIT USED	146
FIGURE 6.7.1: SEPARATING INSPECTION AND UPDATING OF ‘BALANCE’	147
FIGURE 6.7.2: OPTIMISTIC UPDATING OF ‘BALANCE’	148
FIGURE 6.7.3: OPTIMISTIC UPDATING OF ‘STOCK’	149
FIGURE 6.8.1: A FRONT-END/BACK-END DESIGN.....	153
FIGURE 6.9.1: STRUCTURE OF A POSSIBLE CASE TOOL	154
FIGURE 7.3.1: BOSS IS INCOMPATIBLE WITH ITSELF.....	169
FIGURE 8.2.1: USES AND DEFINITIONS OF VARIABLES.....	181
FIGURE 8.2.2: REFERENCE TO AN ARRAY ELEMENT.....	182
FIGURE 8.2.3: DEFINITION OF AN ARRAY ELEMENT	182
FIGURE 8.2.4: DROPPING AN ARRAY ELEMENT.....	182
FIGURE 8.2.5: A DEAD-END DEFINITION	183
FIGURE 8.3.1: STATE TRANSITION DIAGRAM OF AN ACCOUNT	184
FIGURE 8.5.1: CONSTRUCTION FOR A STATEMENT SEQUENCE	186
FIGURE 8.8.1: CONSTRUCTION FOR AN ‘IF’ STATEMENT.....	189
FIGURE 8.11.1: SDG FOR 2 ITERATIONS OF EXAMPLE 8.11.3	195
FIGURE 8.11.2: LEXICAL DEPENDENCES FOR ONE ITERATION OF EXAMPLE 8.11.3.....	195
FIGURE 8.11.3: LEXICAL DEPENDENCES FOR 2 ITERATIONS OF EXAMPLE 8.11.3.....	195
FIGURE 8.11.4: LEXICAL DEPENDENCES FOR 3 ITERATIONS OF EXAMPLE 8.11.3.....	195
FIGURE 8.11.5: AN ALTERNATIVE CONSTRUCTION FOR A ‘WHILE’ LOOP.....	196
FIGURE 8.11.6: THE SDG FOR THE LOOP OF EXAMPLE 8.11.2.....	197
FIGURE 8.11.7: SDG FOR THE LOOP OF EXAMPLE 8.11.4.....	198
FIGURE 9.2.1: THE LATTICE OF PARTITIONS FOR 4 ELEMENTS	202
FIGURE 9.2.2: A CANONICAL MINIMAL PROCESS GRAPH	203
FIGURE 9.2.3: THE LATTICE OF FEASIBLE PROCESS GRAPHS FROM FIGURE 9.2.2	205
FIGURE 9.3.1: AN ACYCLIC SDG	205
FIGURE 9.3.2: A CYCLIC SDG.....	206
FIGURE 9.3.3: THE CANONICAL PROCESS GRAPH OF FIGURE 9.3.1.....	206
FIGURE 9.3.4: THE CANONICAL PROCESS GRAPH OF FIGURE 9.3.2.....	206
FIGURE 9.4.1: AN INFEASIBLE PROCESS GRAPH	207
FIGURE 9.4.2: AN OPTIMAL COMPOSITION OF FIGURE 9.3.3.....	208
FIGURE 9.4.3: A FEASIBLE COMPOSITION OF FIGURE 9.3.4.....	208
FIGURE 9.5.1: A FEASIBLE COMPOSITION APPLIED TO FIGURE 9.3.3.....	210

List of Figures

FIGURE 9.5.2: A FEASIBLE COMPOSITION APPLIED TO FIGURE 9.5.1..... 210

FIGURE 9.5.3: A NOTHER OPTIMAL COMPOSITION OF FIGURE 9.3.3..... 210

FIGURE 9.5.4: A PROCESS GRAPH OFFERING A CHOICE OF COMPOSITIONS 210

FIGURE 9.5.5: THE EFFECT OF COMPOSING TWO UNCONNECTED PROCESSES 211

FIGURE 9.5.6: THE EFFECT OF COMPOSING TWO CONNECTED PROCESSES 211

FIGURE 9.6.1: ANOTHER PROCESS GRAPH OFFERING A CHOICE OF COMPOSITIONS 212

FIGURE 9.6.2: THE OPTIMAL PROCESS GRAPH..... 212

FIGURE 9.6.3: A SUBOPTIMAL PROCESS GRAPH..... 212

FIGURE 9.7.1: PART OF A DECISION TREE..... 215

FIGURE 9.9.1: A PARTITIONED SDG..... 224

FIGURE 10.6.1: BEFORE OPTIMISATION..... 250

FIGURE 10.6.2: AFTER OPTIMISATION..... 250

FIGURE 11.1.1: AN E-R DIAGRAM FOR ORDER PROCESSING..... 254

FIGURE 11.1.2: PROCESS GRAPH FOR ORDERS WITH CREDIT USED..... 256

FIGURE 11.1.3: DATA-FLOW DIAGRAM FOR ORDERS WITH CREDIT USED 257

FIGURE 11.1.4: SYSTEM SPECIFICATION DIAGRAM FOR ORDERS WITH CREDIT USED 261

FIGURE 11.1.5: NETWORK DIAGRAM FOR ORDERS WITH CREDIT USED 262

FIGURE 11.3.1: A MISSILE GUIDANCE SUB-SYSTEM..... 270

FIGURE 11.3.2: THE MISSILE GUIDANCE SDG..... 271

EXAMPLE 1.5.1: A PROGRAM FRAGMENT.....	9
EXAMPLE 2.1.1: TWO KINDS OF EVENT.....	20
EXAMPLE 2.1.2: A SYSTEM READ LOOP.....	20
EXAMPLE 2.1.3: EVENT SPECIFICATIONS.....	20
EXAMPLE 2.3.1: COMPONENT PROCESS PROCEDURES.....	24
EXAMPLE 2.5.1: A SIMPLE LIBRARY SYSTEM	27
EXAMPLE 2.5.2A: LIBRARY COMPONENT SPECIFICATIONS (PART 1).....	28
EXAMPLE 2.5.2B: LIBRARY COMPONENT SPECIFICATIONS (PART 2)	29
EXAMPLE 2.5.2C: LIBRARY COMPONENT SPECIFICATIONS (PART 3).....	29
EXAMPLE 2.5.3A: EXPRESSING INDEPENDENT ACCESS (PART 1)	30
EXAMPLE 2.5.3B: EXPRESSING INDEPENDENT ACCESS (PART 2).....	32
EXAMPLE 2.6.1: SHARED VARIABLES	39
EXAMPLE 2.6.2: USING A DECLARE BLOCK.....	40
EXAMPLE 2.7.1: TRANSFORMATION OF LIBRARY COMPONENT SPECIFICATIONS.....	43
EXAMPLE 3.2.1A: RANDOM ACCESS LIBRARY SYSTEM IN COBOL (PART 1)	49
EXAMPLE 3.2.1B: RANDOM ACCESS LIBRARY SYSTEM IN COBOL (PART 2).....	50
EXAMPLE 3.2.1C: RANDOM ACCESS LIBRARY SYSTEM IN COBOL (PART 3)	51
EXAMPLE 3.3.1A: SEQUENTIAL ACCESS LIBRARY SYSTEM IN COBOL (PART 1)	53
EXAMPLE 3.3.1B: SEQUENTIAL ACCESS LIBRARY SYSTEM IN COBOL (PART 2).....	54
EXAMPLE 3.3.1C: SEQUENTIAL ACCESS LIBRARY SYSTEM IN COBOL (PART 3)	56
EXAMPLE 3.3.2: SEQUENTIAL ACCESS LIBRARY SYSTEM IN COBOL (PART 4).....	57
EXAMPLE 3.3.2B: SEQUENTIAL ACCESS LIBRARY SYSTEM IN COBOL (PART 5)	58
EXAMPLE 3.3.2C: SEQUENTIAL ACCESS LIBRARY SYSTEM IN COBOL (PART 6).....	59
EXAMPLE 4.2.1: CHECKING FOR GREEDY BRANCHES IN A LIBRARY SYSTEM.....	78
EXAMPLE 4.2.2: CHECKING FOR GREEDY BRANCHES — COMPONENT PROCESSES	79
EXAMPLE 4.4.1: AN APPARENT DEPENDENCE	82
EXAMPLE 4.4.2: THE DEPENDENCE REMOVED.....	83
EXAMPLE 4.4.3: APPARENT DEPENDENCE COMPONENTS.....	83
EXAMPLE 4.4.4: RE-USE OF A LOCAL VARIABLE.....	84
EXAMPLE 4.4.5: RE-USE OF A LOCAL VARIABLE: COMPONENTS	84
EXAMPLE 4.4.6: TEMPORARY USE OF A STATE VARIABLE.....	85
EXAMPLE 4.6.1: A BORROW EVENT.....	88
EXAMPLE 4.6.2: A REVISED BORROW EVENT.....	88
EXAMPLE 4.6.3: REMOTE PROCEDURE CALL.....	90
EXAMPLE 4.8.1: DEPTH FIRST SEARCH.....	98
EXAMPLE 4.9.1: A STUDENT RECORD SYSTEM.....	101
EXAMPLE 4.9.2: PROCESS BOUNDARY FOR ‘ENROL’ EVENTS	104
EXAMPLE 4.9.3: COMPONENT PROCESS OF THE STUDENT RECORD SYSTEM	105
EXAMPLE 4.9.4: PARALLEL IMPLEMENTATION OF PROCESS_SQE.....	106
EXAMPLE 4.9.5: PARALLEL IMPLEMENTATION OF PROCESS_A	106
EXAMPLE 5.1.1: CHECKING THE INTEGRITY OF CLASS SIZES.....	111
EXAMPLE 5.1.2: PROCESS SPECIFICATIONS FOR CHECKING CLASS SIZES	112
EXAMPLE 5.2.1: A TRANSACTION BETWEEN TWO ACCOUNTS.....	113
EXAMPLE 5.2.2: COMPONENTS OF A TRANSACTION BETWEEN TWO ACCOUNTS	113

List of Examples

EXAMPLE 5.2.3: A ‘SAFE’ TRANSACTION BETWEEN TWO ACCOUNTS.....	114
EXAMPLE 5.2.4: AN ATTEMPT TO PARALLELISE A ‘SAFE’ TRANSACTION.....	115
EXAMPLE 5.2.5: A ‘CAREFUL’ TRANSACTION BETWEEN TWO ACCOUNTS	116
EXAMPLE 5.2.6A: THE 1ST COMPONENT OF A ‘CAREFUL’ TRANSFER.....	116
EXAMPLE 5.2.6B: THE 2ND COMPONENT OF A ‘CAREFUL’ TRANSFER	117
EXAMPLE 5.4.1: IMPLICIT AGREEMENT.....	123
EXAMPLE 5.4.2: EXPLICIT AGREEMENT.....	123
EXAMPLE 5.4.3: AN AGREEMENT CONFLICT.....	123
EXAMPLE 5.4.4: CONFLICT WITH ‘A’	123
EXAMPLE 5.4.5: CONFLICT WITH ‘B’	123
EXAMPLE 5.4.6: UNDIRECTED INTRANSITIVITY OF CONFLICT	124
EXAMPLE 5.4.7: DIRECTED PSEUDO-TRANSITIVITY OF CONFLICT	124
EXAMPLE 5.6.1: CALCULATING REQUIREMENTS FOR PARTS	129
EXAMPLE 6.1.1: SPECIFYING AN ‘ORDER’	132
EXAMPLE 6.3.1: SPECIFYING ‘OPEN’ ‘CLOSE’ AND ‘PAY’ EVENTS.	136
EXAMPLE 6.4.1: SPECIFYING ‘OPEN’ ‘CLOSE’ AND ‘PAY’ EVENTS.	140
EXAMPLE 6.5.1: SPECIFYING AN ‘ORDER’ WITH A CREDIT LIMIT	142
EXAMPLE 6.6.1: SPECIFYING AN ‘ORDER’ WITH A COMMITMENT.....	143
EXAMPLE 6.6.2: SPECIFYING AN ‘ORDER’ WITH CREDIT USED.....	145
EXAMPLE 6.7.1: SPECIFYING AN OPTIMISTIC FAIL-SAFE ‘ORDER’.....	150
EXAMPLE 6.8.1: CONDITIONAL SPECIFICATION OF THE ‘CLOSE’ EVENT.....	152
EXAMPLE 6.8.2: UNCONDITIONAL SPECIFICATION OF THE ‘CLOSE’ EVENT.	152
EXAMPLE 7.1.1: LATEST PROCESSES FOR ASSIGNMENTS	158
EXAMPLE 7.1.2: EARLIEST PROCESSES FOR EXPRESSIONS	159
EXAMPLE 7.1.3: EARLY ASSIGNMENT OF EXPRESSIONS TO PROCESSES.....	161
EXAMPLE 7.1.4A: THE 1ST EVENT PROCEDURE	161
EXAMPLE 7.1.4B: THE 2ND TO 5TH EVENT PROCEDURES	162
EXAMPLE 7.1.5A: THE 2ND EVENT PROCEDURE.....	162
EXAMPLE 7.1.5B: THE 3RD TO 5TH EVENT PROCEDURES	163
EXAMPLE 7.1.6A: THE 3RD EVENT PROCEDURE.....	163
EXAMPLE 7.1.6B: THE 4TH AND 5TH EVENT PROCEDURES	163
EXAMPLE 7.1.7A: THE 4TH EVENT PROCEDURE.....	163
EXAMPLE 7.1.7B: THE 5TH EVENT PROCEDURE	164
EXAMPLE 7.1.8A: A MODIFIED ORDER OF ASSIGNMENTS	164
EXAMPLE 7.1.8B: A MODIFIED VERSION OF THE 3RD PROCESS	165
EXAMPLE 7.2.1: THE 2ND EVENT PROCEDURE ADAPTED FOR INDEPENDENT ACCESS	166
EXAMPLE 7.2.2A: THE 1ST EVENT PROCEDURE FOR A ‘CAREFUL’ TRANSACTION.....	167
EXAMPLE 7.2.2B: THE 2ND EVENT PROCEDURE FOR A ‘CAREFUL’ TRANSACTION.....	167
EXAMPLE 7.2.3: THE 2ND EVENT PROCEDURE TRANSFORMED FOR INDEPENDENT ACCESS	167
EXAMPLE 7.2.4: THE 1ST EVENT PROCEDURE TRANSFORMED FOR INDEPENDENT ACCESS	168
EXAMPLE 7.3.1: A ‘COMPATIBLE’ WHILE LOOP.....	169
EXAMPLE 7.3.2: AN ‘INCOMPATIBLE’ WHILE LOOP.....	169
EXAMPLE 7.3.3: FORMING A CARTESIAN PRODUCT	170
EXAMPLE 7.3.4: CARTESIAN PRODUCT PROCESSES.....	170

EXAMPLE 7.3.5: INTERACTING LOOPS	171
EXAMPLE 7.3.6: NON-INTERACTING LOOPS	171
EXAMPLE 7.3.7: NON-INTERACTING LOOPS	172
EXAMPLE 7.5.1: FINDING PERCENTAGES OF A TOTAL.....	174
EXAMPLE 7.5.2: USING A LOCAL ARRAY.....	175
EXAMPLE 7.5.3: FINDING AN AVERAGE.....	176
EXAMPLE 7.5.4: LOOP FOLDING.....	176
EXAMPLE 8.11.1: A ‘COMPATIBLE’ WHILE LOOP	193
EXAMPLE 8.11.2: AN ‘INCOMPATIBLE’ WHILE LOOP	193
EXAMPLE 8.11.3: AN INDIRECT DEPENDENCE	194
EXAMPLE 8.11.4: A BADLY SPECIFIED ALL LOOP	198
EXAMPLE 8.11.5: A DEPENDENCE BETWEEN ITERATIONS.....	199
EXAMPLE 10.1.1: SPECIFYING THE LIBRARY SYSTEM.....	230
EXAMPLE 10.1.2: THE SYNTAX TREE OF EXAMPLE 10.1.1.....	231
EXAMPLE 10.2.1: PART OF THE OUTPUT FROM THE ANALYSER.....	235
EXAMPLE 10.2.2: THE INPUT TO THE CANONISER.....	236
EXAMPLE 10.3.1: PART OF THE CANONISER OUTPUT	239
EXAMPLE 10.4.1: PART OF THE OPTIMISER OUTPUT	240
EXAMPLE 10.5.1: PART OF THE OUTPUT FROM THE OPTIMISER.....	248
EXAMPLE 10.5.2: THE OUTPUT OF THE GENERATOR	249
EXAMPLE 10.6.1: A SPECIFICATION WITH A WHILE LOOP	251
EXAMPLE 11.2.1: SPECIFYING AN ‘ORDER’ WITH CREDIT USED	266

List of Tables

TABLE 9.7.1: OUTCOMES AFTER THE FIRST DECISION POINT.....	216
TABLE 9.7.2: OUTCOMES AFTER THE SECOND DECISION POINT.....	217
TABLE 9.7.3: OUTCOMES AFTER THE THIRD DECISION POINT.....	217
TABLE 9.7.4: OUTCOMES AFTER THE 23RD DECISION POINT.....	218
TABLE 9.7.5: OUTCOMES AFTER THE 26TH DECISION POINT.....	218
TABLE 9.7.6: 1ST SOLUTION FOUND BY BRANCH-AND-BOUND.....	219
TABLE 9.7.7: 2ND SOLUTION FOUND BY BRANCH-AND-BOUND.....	219
TABLE 9.8.1: AFTER THE 1ST DECISION.....	220
TABLE 9.8.2: AFTER THE 2ND DECISION.....	221
TABLE 9.8.3: AFTER THE 3RD DECISION.....	221
TABLE 9.8.4: AFTER THE 4TH DECISION.....	221
TABLE 10.1.1: SYNTAX OF THE SPECIFICATION LANGUAGE.....	229