
Contents

Contents	vii
List of Figures	xiii
Author’s Preface	xvii
Acknowledgments	xix
1 Introduction	1
1.1 Computation, Traditionally	1
1.2 Plans, Activities, Platforms	2
1.3 Limitations of Traditional Sequential Computing	3
1.4 Why Sequential Processors Run Hot	6
1.5 Scalability & Speedup	7
1.6 Complexity and Decomposition	10
1.7 Some Traditional Approaches to Concurrency	11
1.7.1 Programming Models	11
1.7.2 Pitfalls of Traditional Concurrency	14
1.7.3 Dealing with Latency	15
1.8 Scope, Goals and Directions of This Book	16
1.9 History of This Approach	18
1.10 Disclaimer	20
2 Activities and Resources	21
2.1 Introduction	21
2.1.1 Plans and Activities	21
2.1.2 Resources	22
2.1.3 Permissions	23
2.2 Resources	24
2.2.1 Control State	25
2.2.2 Using and Manipulating Control State	27
2.2.3 Mutually Exclusive Access to Resources	28
2.2.4 Waffles Example	29

2.3	Delineating Activities from Their Contexts	31
2.3.1	Roles and Transitions	32
2.3.2	Usages vs. Permissions	33
2.3.3	Plan Interface	34
2.3.4	Plan/Context Conformance	35
2.4	Strategies Are Plans	38
2.5	Representing Strategies as Text: SPLAT	38
2.6	Summary	40
3	Fundamental Activities: Actons	43
3.1	Activity Properties	43
3.1.1	Determinism	46
3.1.2	Serializability	48
3.1.3	Safety, Liveness, and Fairness	52
3.2	Proposal for Actons and Tactics	56
3.2.1	The Basics	56
3.2.2	Liveness and Fairness	58
3.2.3	Some Common Representations for Tactics	59
3.2.4	Termination (\perp Transitions)	61
3.2.5	Beltedness	62
3.2.6	Predictability (Concurrent Access to Resources)	62
3.2.7	Summary of Acton/Tactic Definition	63
3.3	Examples	63
3.3.1	Waffles	64
3.3.2	Software Subroutines as Tactics	65
3.3.3	Natural Events as Actons	67
3.3.4	ACID Transactions	68
3.4	Summary	69
4	Predictability: Concurrent Observers, Buffers, & Broadcast	71
4.1	Common approaches to shared access	71
4.1.1	Concurrent Observers	72
4.1.2	Broadcast/Multicast	72
4.1.3	Throttling and Buffering	73
4.2	Predictability	74
4.2.1	Snapshots: An Implication of Predictability	74
4.2.2	Predictability as “Wrinkle in Time”	76
4.2.3	Optimizing Snapshots	77
4.3	Revisiting the Scenarios, in Light of Predictability	78
4.3.1	Throttling and Buffering	78
4.3.2	Multicast	80
4.3.3	Concurrent Observers and/or Broadcast	80
4.3.4	Potential Oversimplifications	81
4.4	Summary	82
5	Making It Go: The Director	83
5.1	Introduction	83
5.2	Duties of the Director	84

5.2.1	Semantics	85
5.2.2	Embedding/Mapping	91
5.2.3	Scheduling	92
5.2.4	Communication	94
5.3	Example of a Simple Director	98
5.3.1	Embedding (Static)	98
5.3.2	Content State Movement	100
5.3.3	Finding Content State	102
5.4	Remaining Director Duties	103
5.5	Summary	104
6	Encapsulation: Building More Complex Activities	107
6.1	Issues with Interfacing Strategies	107
6.1.1	Breaking Existing Plans Down into Smaller Plans	108
6.1.2	Building Existing Plans Up into Bigger Plans	110
6.1.3	Activity Lifetime	111
6.2	Forming Activateable Plans from Strategies	112
6.2.1	Formal Resources: Roles for Strategies	112
6.2.2	Strategy Interfaces: Resources as Finite State Machines (FSMs)	115
6.2.3	Predictable Roles Revisited	117
6.3	Finite-lived Composite Activities	119
6.3.1	Finishing an Activity	119
6.3.2	“Cleaning Up” an Activity: Garbage Collection	120
6.4	Summary	121
7	Constants, Repositories, Activation, and Recursion	123
7.1	Constants	123
7.2	Repositories	124
7.2.1	Introduction	124
7.2.2	Constant Paths (Repositories as Constants)	125
7.2.3	Resolution of Constants (Time and Source)	125
7.2.4	Resolving in Parent Repositories	126
7.2.5	Constant recursion	127
7.2.6	Content Domains as Types	127
7.3	Activation: How Contexts Really Work	129
7.3.1	\$plan Role Binding	129
7.3.2	Delaying/Sensing Plan Activations	131
7.3.3	Recursion: Activating a Plan within the Same Plan	132
7.3.4	The Root Plan and Root Context	133
7.4	Summary	134
8	Revisiting Computation	137
8.1	Introduction	137
8.2	Sequential Computation	138
8.2.1	Plans and Computation	138
8.2.2	Example of TDSP and its Computation: Binary Search	138
8.2.3	Computations as Functions	140

8.2.4	Example of Function Corresponding to Computation . . .	143
8.2.5	Structured Planning	144
8.2.6	Summary: Sequential Algorithms and Computation . . .	144
8.3	Concurrent Computation	145
8.3.1	Algorithms, Programs, Computation	145
8.3.2	Example of Concurrent Computation	147
8.3.3	Characterizing Input and Output for Strategies	149
8.3.4	Computations as Functions	150
8.3.5	Example of function corresponding to computation	151
8.3.6	Structured Planning	154
8.4	Summary	154
9	Nondeterminism and Atomicity	157
9.1	Nondeterminism	157
9.1.1	Definition	157
9.1.2	Syntactic (Strategy) Computation Determinism	158
9.1.3	Semantic (Tactic) Computation Determinism	160
9.1.4	Result Determinism (Specifically, Semideterminism) . . .	161
9.1.5	Re-creating Computations	165
9.1.6	Other Kinds of Nondeterminism	166
9.2	Atomicity	167
9.2.1	Definition	167
9.2.2	Atomicity Using an Atomic “Start” Action	169
9.2.3	Atomicity Using Activation Formal Resources	171
9.2.4	The Downside of Atomicity	172
9.3	Summary	172
10	Axiomatic Semantics	175
10.1	Intro	175
10.2	F-Net Syntax	177
10.3	F-Net Semantics	179
10.3.1	Initialization Axiom	180
10.3.2	Time Axiom	181
10.3.3	Atomicity Axiom	181
10.3.4	Safety Axiom	181
10.3.5	Liveness Axiom	181
10.3.6	Firing Axiom	181
10.4	Theorems	183
10.4.1	Nodes in Computation Are Partially Ordered	183
10.4.2	States for a Resource in Computation Are Totally Ordered	183
10.4.3	Computations as Functions	185
10.4.4	Tracing a Computation	186
10.4.5	Computations with Identical Resource Traces are Isomor- phic	188
10.5	Summary	192
11	Sets	195
11.1	Motivation	195

11.2	Paths and Formal Resource Sets	197
11.3	Role Sets and Splice Bindings	198
11.4	Bundlings	199
11.5	Wildcards	199
11.6	Resource Sets	200
11.7	Summary	201
12	Objects	203
12.1	Intro to Object-Oriented Planning	203
12.1.1	Outline of (Some) OO Principles	204
12.1.2	Strategies as Classes and Instances (Objects)	206
12.2	Encapsulated State	207
12.3	Partial Binding	212
12.3.1	Binding Constraints	213
12.3.2	Methods	214
12.4	Delegation	215
12.5	Class Structure (Subtyping)	219
12.5.1	Strategies as Content Domains	219
12.5.2	Superclass Designation	220
12.5.3	Abstract Classes	221
12.6	Example	222
12.7	Summary	227
13	Arrays	229
13.1	Resource Arrays	230
13.1.1	Form and Terminology	230
13.1.2	Default Binding, Bindall	232
13.1.3	Bindany	232
13.2	Binding Modifiers	233
13.2.1	Reduction	237
13.2.2	Selection	238
13.2.3	Translation	239
13.2.4	Mapping	241
13.2.5	Permutation	243
13.2.6	Combining Binding Modifiers	245
13.2.7	Alternate Representations for Binding Modifiers	247
13.3	Delimited and Undelimited Resource Arrays	247
13.3.1	Establishing Array Size: Delimiters	249
13.3.2	Sensing Array Size	249
13.4	Example: Bounded Buffer Object (Queue, FIFO)	250
13.4.1	The Plan	251
13.4.2	Analysis	252
13.5	Summary	253
14	Data Parallelism	255
14.1	Creating Contexts Dynamically	255
14.1.1	Basic Function of Dup	256
14.1.2	Number of Replicants	256

14.1.3	Behavior of Role Bindings	257
14.1.4	Combining Dups	258
14.2	Using dups with binding modifiers for data parallelism	260
14.3	Examples	263
14.3.1	Linked List Object	263
14.3.2	Matrix Multiply	268
14.3.3	Matrix Multiply Variations	271
14.3.4	Quicksort	276
14.4	Embedding/Scheduling Data Parallel Plans	282
14.5	Summary	283
15	Engineering Correct Plans	285
15.1	Introduction	285
15.1.1	Correctness of Tactics	285
15.1.2	Correctness of Strategies	286
15.1.3	Specifications for strategies	288
15.2	Refactoring	289
15.2.1	Refactoring Control State	290
15.2.2	Refactoring Tactics	292
15.3	Related Formal Models	292
15.3.1	UNITY/Guarded Commands	293
15.3.2	Petri Nets	293
15.3.3	Turing Machines	294
15.3.4	Chemical Abstract Machine (CHAM)	297
15.3.5	Other Models	297
16	The Future: What If?	299
16.1	Publishing More Existing Research	299
16.2	Making “Scalable Planning” Simply “Planning”	300
16.3	Creating a ScalPL-Friendly World	301
16.4	Rethinking Apps	302
16.5	Conclusion	303
Index		305