

Contents

Abstract	vii
1 Introduction	1
1.1 Prototyping Parallel Algorithms	1
1.2 A Look at Software Construction	3
1.2.1 Informal and Formal Specifications	3
1.2.2 Formal Specifications and Prototyping	6
1.3 Overview	7
1.4 Acknowledgments	7
I Setting	9
2 Prototyping	11
2.1 Goals of Prototyping	11
2.2 Approaches to Prototyping	13
2.3 Summary	14
3 Parallel Programming	15
3.1 Goals of Parallel Programming	15
3.2 Approaches to Parallel Programming	17
3.2.1 Data Parallelism	19
3.2.2 Parallel Object-Oriented Programming	20
3.2.3 Parallel Functional Programming	22
3.2.4 Parallel Logic Programming	23
3.2.5 Unbounded Nondeterministic Iterative Transformations	25
3.2.6 Multiset Transformations	27
3.2.7 Virtual Shared Memory	28
3.2.8 The Shared Data-Object Model	29
3.2.9 Generative Communication	31
3.2.10 Evaluation	35
3.3 Summary	40

II	Generative Communication in Set-Oriented Prototyping	41
4	The Prototyping Language ProSET	43
4.1	Data Structures	43
4.2	Control Structures	45
4.3	Introductory Examples	45
4.4	Exception Handling	47
4.5	Persistence	47
4.6	Modules	49
4.7	Summary	49
5	Informal Semantics of ProSET-Linda	51
5.1	Process Creation	51
5.2	Program and Process Termination	53
5.3	Handling Multiple Tuple Spaces	54
5.4	Tuple-Space Operations	55
5.4.1	Depositing Tuples	55
5.4.2	Fetching Tuples	57
5.4.3	Meeting Tuples	59
5.4.4	Nondeterminism and Fairness while Matching	61
5.5	Summary	61
6	Example Programs	63
6.1	A Master-Worker Application with Limited Tuple Spaces	63
6.2	The Queens' Problem Revisited	65
6.3	Parallel Matrix Multiplication	65
6.4	The Traveling Salesman Problem	65
6.5	The Dining Philosophers Problem	70
6.6	Summary	72
7	Discussion of Other Approaches to Generative Communication and Some Design Alternatives	73
7.1	Shortcomings of C-Linda	74
7.2	Comparison of ProSET's and C-Linda's Tuple-Space Operations	74
7.3	Customized Matching	75
7.4	Selective Matching	75
7.5	Aggregate and Accumulative Matching	76
7.6	Limited Tuple Spaces and Non-blocking Matching	76
7.7	Fair Matching	77
7.8	Extending the Type System for Matching	78
7.9	Formals and Templates with First-Class Rights	78

7.10	Update Operations on Tuples in Tuple Space	79
7.11	Process Creation and Process Identities	79
7.12	Multiple Tuple Spaces and Tuple-Space Identities	80
7.13	Data Parallelism in Generative Communication	82
7.14	Fault-tolerance	83
7.15	Persistence	84
7.16	Overloading Predefined Operators	84
7.17	Summary	85
8	Formal Semantics of PROSET-Linda	87
8.1	Basic Definitions	87
8.1.1	Abstractions for the Embedding into the Computation Language	87
8.1.2	Types and Values	88
8.1.3	Tuples	89
8.1.4	Formals and Templates	90
8.2	Matching	92
8.3	Tuple Spaces	92
8.4	Programs and Processes	94
8.5	Exceptions	97
8.6	Handling Multiple Tuple Spaces	97
8.6.1	CreateTS	98
8.6.2	ExistsTS	99
8.6.3	ClearTS	100
8.6.4	RemoveTS	101
8.6.5	The Tuple-Space Library	101
8.7	Tuple-Space Operations	102
8.7.1	Some Preliminary Definitions	102
8.7.2	Depositing Tuples	106
8.7.3	Fetching Tuples	107
8.7.4	Meeting Tuples	109
8.7.5	Fairness of the Tuple-Space Operations	111
8.8	Program Execution	112
8.9	Summary	113

III	Implementation	115
9	Refining the Formal Specification	117
9.1	Semantics versus Implementation Design	117
9.2	Data Refinement	118
9.3	Operation Refinement	119
9.3.1	Some Preliminary Definitions	122
9.3.2	Depositing Tuples	124
9.3.3	Fetching Tuples	124
9.3.4	Meeting Tuples	126
9.4	Correctness of the Refinement	126
9.4.1	Modeling Bags by Sequences	127
9.4.2	Satisfaction of the Fairness Properties	128
9.5	Summary	129
10	A Prototype Implementation	131
10.1	The Compiler	131
10.2	The Tuple-Space Management	132
10.2.1	Handling Multiple Tuple Spaces	132
10.2.2	Tuple-Space Operations	134
10.3	Summary	140
11	Some General Issues for Implementations of PROSET-Linda	141
11.1	Some Existing Implementations of Linda Variants	141
11.2	Implementation Techniques	142
11.3	Optimizations	144
11.3.1	Partitioning the Tuple Space	144
11.3.2	In-place Updates	145
11.3.3	Hardware Support	145
11.3.4	Data Structure Selection	145
11.4	The Unpredictable Performance of Linda	146
11.5	Summary	146
IV	Résumé and Outlook	147
12	Résumé	149
13	Outlook	151
13.1	Extending Matching to Unification	151
13.2	An Implementation of a Graphical Debugger	152
13.3	An Implementation on a Local Area Network	153
13.4	Optimizing Analysis of Tuple-Space Access	153

V	Appendices	155
A	The Abstract Grammar for the Tuple-Space Operations	157
B	The Specification Language Object-Z	159
B.1	Schemas	160
B.2	Axiomatic Descriptions	161
B.3	Generic Descriptions	161
B.4	Free Type Definitions	161
B.5	Expressions	162
B.6	Classes	162
B.7	Remarks Concerning the Usability of Z and Object-Z	164
C	Types of All Names Defined Globally	165
Indices		173
Index of Formal Definitions		173
Index of Explained Object-Z Symbols and Keywords		175
General Index		177
Bibliography		181