Tag-free Garbage Collection Using Explicit Type Parameters p. 1
Garbage Collection for Strongly-Typed Languages using Run-time Type Reconstruction p. 12
Collecting More Garbage p. 24
Concurrent Replicating Garbage Collection p. 34
Characterisation of Object Behaviour in Standard ML of New Jersey p. 43
Static Dependent Costs for Estimating Execution Time p. 65
Using Run-Time Sizes of Data Structures to Guide Parallel-Thread Creation p. 79
Locality, Causality and Continuations p. 91
Syntax and Semantics of a Persistent Common Lisp p. 103
Talking about Modules and Delivery p. 113
Sweet Harmony: The Talk/C++ Connection p. 121
Lambda, the Ultimate Label, or a Simple Optimizing Compiler for Scheme p. 128
An Efficient Implementation of Multiple Return Values in Scheme p. 140
Space-Efficient Closure Representations p. 150
Using Hindley-Milner Type Inference to Optimise List Representation p. 162
A Randomized Implementation of Multiple Functional Arrays p. 173
Unrolling Lists p. 185
A Direct Algorithm for Type Inference in the Rank-2 Fragment of the Second-Order I-Calculus p. 196
Semantics of Type Classes Revisited p. 208
Complete Proof Systems for Algebraic Simply-Typed Terms p. 220
Continuation-Based Partial Evaluation p. 227
Type Reconstruction for Variable-Arity Procedures p. 239
A Practical Soft Type System for Scheme p. 250
Parallel Destructive Updating in Strict Functional Languages p. 263
Towards Better Inlining Decisions Using Inlining Trials p. 273
In-place Updates in the Presence of Control Operators p. 283
Analyzing Stores and References in a Parallel Symbolic Language p. 294
Set-Based Analysis of ML Programs p. 306
An Equational Framework for the Flow Analysis of Higher-Order Functional Programs p. 318

Table of Contents provided by Blackwell's Book Services and R.R. Bowker. Used with permission.