1 **Introduction**
- 1.1 Intellectual Precursors **4**
- 1.2 Acknowledgments **4**
- 1.3 References **5**

2 **Overview**
- 2.1 Structure of Classes and Instances **7**
- 2.2 Inheriting Variables and Methods **8**
- 2.3 Data Oriented Programming Using Active Values **9**
- 2.4 Knowledge Bases **10**

3 **Creating and Using Objects**
- 3.1 Sending a Message to an Object **13**
- 3.2 Creating a New Instance **14**
- 3.3 Naming and Pointing to Objects **14**
- 3.4 De ning a New Class **15**
- 3.5 De ning a Method **16**

4 **Object Variables and Properties**
- 4.1 Access Expressions **18**
- 4.2 Getting Variable and Property Values **19**
- 4.3 Putting Variable Values and Property Values **20**
- 4.4 Non- triggering Get and Put **21**
- 4.5 Local Get Functions **21**
- 4.6 Accessing Class and Method Properties **22**
- 4.7 General Get and Put Functions **23**
- 4.8 Summary of Get and Put Functions **24**

5 **Active Values**
- 5.1 Active Values Notation **25**
- 5.2 Nested Active Values **25**
- 5.3 Active Values as Default Values **26**
- 5.4 Standard Access Functions **26**
- 5.5 User- De ned Access Functions **27**

6 **Combining Inherited Methods**
- 6.1 Augmenting an Inherited Method **31**
- 6.2 Combining Multiple Inherited Methods **32**
- 6.3 General Method Invocation **32**

7 **Instance Creation**
- 7.1 Specifying Values at Instance Creation **34**
- 7.2 Sending a Message at Instance Creation **34**
- 7.3 Computing a Value at First Fetch **35**
- 7.4 Computing a Value at Instance Creation **35**
- 7.5 Special Actions at Instance Creation **36**

8 **Composite Objects**
8.1 Basic Concepts for Composite Objects **38**
8.2 Specializing Composite Objects **39**
8.3 Conditional and Iterative Templates **40**

9 **Loops Knowledge Bases**
9.1 Review of Knowledge Base Concepts **41**
9.2 Environmental Objects and Boot Layers **42**
9.3 Starting With No Preexisting Knowledge Bases **44**
9.4 Continuing from a Previous Session **45**
9.5 Starting from a Community Knowledge Base **46**
9.6 Freezing and Thawing References to Knowledge Bases **47**
9.7 Using Several Knowledge Bases in an Environment **48**
9.8 Changing the Associations of Objects **49**
9.9 Switching Among Environments **49**
9.10 Saving Parts of a Session **51**
9.11 Copying Layers from one Knowledge Base to Another **51**
9.12 Summarizing and Combining Knowledge Bases **52**
9.13 Subdividing a Knowledge Base **53**
9.14 Going Back to a Previous Boot Layer of a Knowledge Base **54**
9.15 A ecting what is Saved **54**
9.15.1 Temporary Objects **55**
9.15.2 Not Saving some IV values **55**
9.15.3 Ignoring changes on an IV **55**
9.15.4 Getting rid of objects explicitly **56**
9.16 Examining Environmental Objects **56**
9.17 The Class KBState **57**
9.18 The Class KB **58**
9.19 The Class Environment **59**
9.20 The Class Layer **61**
9.21 The Class KBMeta **61**
9.22 The Class EnvironmentMeta **62**

10 **Introduction to Rule-Oriented Programming in LOOPS**
10.1 Introduction **63**
10.2 Basic Concepts **64**
10.3 Organizing a Rule-Oriented Program **65**
10.4 Control Structures for Selecting Rules **66**
10.5 One-Shot Rules **68**
10.6 Task-Based Control for RuleSets **69**
10.7 Control Structures for Generators **71**
10.8 Saving an Audit Trail of Rule Invocation **72**
10.8.1 Motivations and Applications **72**
10.8.2 Overview of Audit Trail Implementation **73**
10.8.3 An Example of Using Audit Trails **73**
10.9 Comparison with other Rule Languages **75**
10.9.1 The Rationale for Factoring Meta-Level Syntax **75**
10.9.2 The Rationale for RuleSet Hierarchy **76**
10.9.3 The Rationale for RuleSet Control Structures **76**

10.9.4 The Rationale for an Integrated Programming Environment **78**

11 **The Rule Language**
- 11.1 Rule Forms **80**
- 11.2 Kinds of Variables **81**
- 11.3 Rule Forms **83**
- 11.4 In x Operators and Brackets **83**
- 11.5 Interlisp Functions and Message Sending **85**
- 11.6 Variables and Properties **86**
- 11.7 Perspectives **87**
- 11.8 Computing Selectors and Variable Names **87**
- 11.9 Recursive Compound Literals **88**
- 11.10 Assignment Statements **88**
- 11.11 Meta- Assignment Statements **89**
- 11.12 Push and Pop Statements **90**
- 11.13 Invoking RuleSets **90**
- 11.14 Transfer Calls **91**
- 11.15 Task Operations **91**
- 11.16 Stop Statements **92**

12 **Using Rules in LOOPS**
- 12.1 Creating RuleSets **94**
- 12.2 Editing RuleSets **94**
- 12.3 Copying RuleSets **95**
- 12.4 Saving RuleSets on LISP Files **95**
- 12.5 Printing RuleSets **96**
- 12.6 Running RuleSets from Loops **96**
- 12.7 Installing RuleSets as Methods **96**
- 12.8 Installing RuleSets in ActiveValues **97**
- 12.9 Tracing and Breaking RuleSets **98**
- 12.10 The Rule Exec **99**
- 12.11 Auditing RuleSets **99**

13 **Using the Loops System**
- 13.1 Starting up the System **101**
- 13.2 The Loops Screen Setup **101**
- 13.3 Using the Browser **102**
  - 13.3.1 Using the Class Browser **102**
  - 13.3.2 Building Your Own Browser **105**
- 13.4 Editing in Loops **108**
  - 13.4.1 Editing a Class **108**
  - 13.4.2 Editing an Instance **109**
  - 13.4.3 Editing a Method **110**
- 13.5 Inspecting in Loops **110**
  - 13.5.1 Inspecting Classes **110**
  - 13.5.2 Inspecting Instances **110**
- 13.6 Errors in Loops **111**
  - 13.6.1 When the Object is Not Recognized **111**
  - 13.6.2 When the Selector is Not Recognized **111**

13.7 Breaking and Tracing Methods **112**
13.8 Monitoring Variable Access **112**

14 **The LOOPS Kernel**
14.1 The Golden Braid (Object, Class, MetaClass) **113**
14.2 Perspectives and Nodes **113**
14.3 Useful Mixins **114**
14.4 The MetaClass Named "Class" **115**
14.5 The Class Named "Object" **118**
14.6 Functions for changing Loops Structure **120**
14.6.1 Moving and Renaming Methods **120**
14.6.2 Moving and Renaming Variables **121**

15 **Loops and the Interlisp System**
15.1 Saving Class and Instance De nitions on Files **122**
15.2 Classes for Lisp Datatypes **122**
15.3 Some Details of the Loops implementation **122**