

Table of Contents

Introduction	1
-------------------------------	---

Part I. Correctness Debugging

1. An Assertion Language for Constraint Logic Programs	
Germán Puebla, Francisco Bueno, and Manuel Hermenegildo	23
1.1 Introduction	23
1.2 Assertions in Program Validation and Debugging	27
1.3 Assertion Schemas for Execution States	29
1.3.1 An Assertion Schema for Success States	30
1.3.2 Adding Preconditions to the Success Schema	31
1.3.3 An Assertion Schema for Call States	32
1.3.4 An Assertion Schema for Query States	33
1.3.5 Program-Point Assertions	34
1.4 Logic Formulae about Execution States	35
1.5 An Assertion Schema for Declarative Semantics	37
1.6 Assertion Schemas for Program Completeness	38
1.7 Status of Assertions	40
1.8 An Assertion Schema for Computations	44
1.8.1 Logic Formulae about Computations	45
1.9 Defining Property Predicates	46
1.9.1 Declaring Property Predicates	47
1.9.2 Defining Property Predicates for Execution States	48
1.9.3 Defining Property Predicates for Computations	50
1.9.4 Approximating Property Predicates	51
1.10 Syntax of and Extensions to the Assertion Language	52
1.10.1 Syntax of the Assertion Language	53
1.10.2 Grouping Assertions: Compound Assertions	54
1.10.3 Some Additional Syntactic Sugar	56
1.11 Discussion	58
References	59

2. A Generic Preprocessor for Program Validation and Debugging	
Germán Puebla, Francisco Bueno, and Manuel Hermenegildo	63
2.1 Introduction	63
2.1.1 Design of the Preprocessor	64
2.1.2 Chapter Outline	68
2.2 Architecture and Operation of the Preprocessor	69
2.2.1 The Syntax Checker	69
2.2.2 The Static Analysers	72
2.2.3 Consistency of the Analysis Results	73
2.2.4 The Assertion Normaliser	74
2.2.5 The Assertion Comparator	75
2.2.6 Assertions for System Predicates	75
2.2.7 Assertions for User-Defined Predicates	77
2.3 Compile-Time Checking	79
2.4 Run-Time Checking	85
2.4.1 Evaluating Atomic Logic Formulae	86
2.4.2 A Program Transformation for Assertion Checking . . .	89
2.5 Customising the Preprocessor for a CLP System:	
The CiaoPP and CHIPRE Tools	93
2.5.1 Describing Built-Ins Using Assertions	94
2.5.2 System Dependent Code for Run-Time Checking	97
2.6 A Sample Debugging Session with the Ciao System	98
2.7 Some Practical Hints on Debugging with Assertions	103
References	104
3. Assertions with Constraints for CLP Debugging	
Claude Läi	109
3.1 Introduction	109
3.2 Form of Assertions	110
3.2.1 Syntax and Meaning of Assertions	110
3.2.2 The Basic Constructs	111
3.3 Correctness Proofs	112
3.3.1 Implementation	112
3.3.2 Example of a Verification	113
3.3.3 Incompleteness Introduced by the Solvers	114
3.3.4 Full Example	115
3.3.5 Samples of Compilations	116
3.4 Run-Time Checking	118
3.5 Conclusion	119
References	120

4. Locating Type Errors in Untyped CLP Programs

Włodzimierz Drabent, Jan Małuszyński, and Paweł Pietrzak	121
4.1 Introduction	121
4.2 The Specification Language	124
4.2.1 Calls and Successes of a CLP Program	124
4.2.2 Describing Sets of Constrained Atoms	126
4.3 An Example Diagnosis Session	128
4.4 The Diagnosis Method	133
4.4.1 Correct and Incorrect Clauses	134
4.4.2 Incorrectness Diagnosis	137
4.5 Delays	138
4.6 The Diagnosis Tool	141
4.7 Limitations of the Approach	143
4.8 Related Work	146
4.9 Conclusions and Future Work	148
References	149

5. Declarative Diagnosis in the CLP Scheme

Alexandre Tessier and Gérard Ferrand	151
5.1 Introduction	151
5.2 Basic Notions of Symptom and Error	154
5.3 Connection between Symptom and Error via Proof-Trees	156
5.4 Diagnosis Algorithm	160
5.5 Abstract Proof-Trees	163
5.6 Implementation	167
5.7 A Diagnosis Session	170
5.8 Conclusion	172
References	173

Part II. Performance Debugging

6. Visual Tools to Debug Prolog IV Programs

Pascal Bouvier	177
6.1 Introduction	177
6.2 Presentation of the Execution Tree Viewer	178
6.2.1 The Box Model	178
6.2.2 Execution Trees	179
6.2.3 The Prolog IV Execution Tree Viewer	183
6.2.4 Textual Information in the Canvas	184
6.3 Viewer Functions Description	184
6.3.1 Colouring Boxes	185
6.3.2 Replay Mode	186

6.3.3	“Replay-Left Mode” Option	186
6.3.4	“Show all Tries” Option	187
6.3.5	Miscellaneous	187
6.4	Working with the Debugger	187
6.4.1	Setting a Temporary Break-Point	188
6.4.2	Replaying the Execution	188
6.5	About Implementation	189
6.6	Conclusion	190
	References	190
7.	Search-Tree Visualisation	
	Helmut Simonis and Abder Aggoun	191
7.1	Introduction	191
7.2	Related Work	192
7.3	Principles of Operation	193
7.3.1	Program Structure	193
7.3.2	Symptoms	193
7.3.3	Operating Mode	194
7.3.4	System Requirements	195
7.4	Interface	195
7.5	Views	197
7.5.1	Types of Views	197
7.5.2	Tree View	198
7.5.3	Variable Views	201
7.5.4	Constraint Views	203
7.5.5	Propagation View	205
7.6	Current State and Further Development	206
7.7	Conclusion	207
	References	208
8.	Towards a Language for CLP Choice-Tree Visualisation	
	Christophe Aillaud and Pierre Deransart	209
8.1	Introduction	209
8.2	CLP: Syntax, Semantics and State Properties	211
8.2.1	Syntax	211
8.2.2	Constraint Domains	212
8.2.3	Constrained Predication and <i>D</i> -Atom	212
8.2.4	Operational Semantics	213
8.2.5	CLP Search-Tree	214
8.2.6	Labelling	216
8.2.7	State Properties	217
8.3	Tree Pruning and CLP Choice-Tree	219
8.3.1	Unique Minimal Pruned Tree	219
8.3.2	Minimal Pruned Tree Construction	221

8.3.3	CLP Choice-Tree	223
8.4	A Language to Specify Views	224
8.4.1	Node Selection	225
8.4.2	Arc Selection	225
8.4.3	Node and Arc Selection Using Differential Information	226
8.4.4	View Specification	227
8.5	Implementation and Examples	227
8.5.1	Displaying the Full Concrete Choice-Tree	228
8.5.2	A General Criterion for a More Synthetic View	229
8.5.3	View of the Labelling	230
8.5.4	Use of State Properties	232
8.5.5	Comparison of Labelling Strategies	232
8.6	Conclusion	234
	References	236

9. Tools for Search-Tree Visualisation: The APT Tool

	Manuel Carro and Manuel Hermenegildo	237
9.1	Introduction	237
9.2	Visualising Control	238
9.3	The Programmed Search as a Search Tree	239
9.4	Representing the Enumeration Process	240
9.5	Coupling Control Visualisation with Assertions	241
9.6	The APT Tool	242
9.7	Event-Based and Time-Based Depiction of Control	245
9.8	Abstracting Control	248
9.9	Conclusions	250
	References	251

10. Tools for Constraint Visualisation: The VIFID/TRIFID Tool

	Manuel Carro and Manuel Hermenegildo	253
10.1	Introduction	253
10.2	Displaying Variables	254
10.2.1	Depicting Finite Domain Variables	254
10.2.2	Depicting Herbrand Terms	258
10.2.3	Depicting Intervals or Reals	258
10.3	Representing Constraints	259
10.4	Abstraction	263
10.4.1	Abstracting Values	263
10.4.2	Domain Compaction and New Dimensions	264
10.4.3	Abstracting Constraints	268
10.5	Conclusions	270
	References	270

11. Debugging Constraint Programs by Store Inspection	
Frédéric Goulard and Frédéric Benhamou	273
11.1 Introduction	273
11.2 Constraint Logic Programming in a Nutshell	274
11.3 Arising Difficulties when Debugging Constraint Programs	276
11.3.1 Constraint Programming Efficiency	276
11.3.2 Drawbacks of CP Expressiveness	278
11.4 Visualising the Store	278
11.4.1 Structuring the Store	279
11.4.2 S-Boxes	281
11.5 Presentation of the Debugger Prototype	285
11.6 Implementation of the S-Box Based Debugger	287
11.6.1 Modification of the Backtracking Process	291
11.6.2 Modification of the Propagation Process	291
11.6.3 Handling the S-Boxes	293
11.7 Conclusion	294
References	296
12. Complex Constraint Abstraction: Global Constraint Visualisation	
Helmut Simonis, Abder Aggoun, Nicolas Beldiceanu, and Eric Bourreau	299
12.1 Introduction	299
12.2 Global Constraint Concepts	301
12.2.1 Cumulative	301
12.2.2 Diffn	302
12.2.3 Cycle	302
12.2.4 Among	303
12.3 Principles of Operation	303
12.3.1 Class Structure	303
12.3.2 Callbacks	305
12.3.3 API	306
12.4 Interface to Search-Tree Tool	307
12.5 Use Outside Search-Tree Tool	308
12.6 Cumulative Visualisers	308
12.6.1 Cumulative Resource	308
12.6.2 Bin Packing	311
12.7 Diffn Visualisers	311
12.7.1 Placement 2D	311
12.7.2 Placement Remains	311
12.8 Cycle Visualiser	313
12.8.1 Geographical Tour	313
12.8.2 Graph Lines	314
12.9 Interaction between Visualisers	315

12.10 Conclusions	315
References	316

Part III. Test Cases

13. Using Constraint Visualisation Tools

Helmut Simonis, Trijntje Cornelissens, Véronique Dumortier, Giovanni Fabris, F. Nanni, and Adriano Tirabosco	321
13.1 Introduction	321
13.2 Debugging Scenarios	322
13.2.1 Finding New Variable/Value Orderings	323
13.2.2 Comparing Two Heuristics	328
13.2.3 Adding Redundant Constraints	331
13.2.4 Discovering Structure in the Problem	334
13.2.5 Identifying Weak Propagation	336
13.3 Case Studies from Industry	337
13.3.1 Scheduling Application Involving Setup Cost	337
13.3.2 Tank Scheduling Application	340
13.3.3 Layout of Mechanical Objects: Graph Colouring	344
13.3.4 Layout of Mechanical Objects: Physical Layout	348
13.4 Analysis and Possible Improvements	351
13.4.1 Need for a Debugging Methodology	351
13.4.2 Program Improvement Due to the Use of Debugging Tools	351
13.4.3 General Conclusions on the CHIP Graphical Debugging Tools	352
13.4.4 Suggestions for Extensions to the CHIP Debugging Tools	353
13.5 Conclusions	355
References	356
Author Index	357
Subject Index	359