

Index

- $\Pi(\mathcal{SD})$ (ASP encoding of \mathcal{AL} system description), 172
- Π^S (reduct), 26
- $\Pi_c(\mathcal{SD})$, 168
- Σ (signature), 11
- σ_0 (initial state), 162
- σ_n (final state), 163
- \mathcal{SD} (system description), 167
- $\mathcal{T}(\mathcal{SD})$ (transition diagram), 167

- abductive support, **105–108**
- aces in succession example, 250–251
- action languages, **165**
- agent, **1**
- agent action, **216**
- agent architecture, **2**
- agent loop, **3**
- \mathcal{AL} , 165–187
 - statements of, 167
- algorithm
 - for answering queries, 148–149
 - for computing answer sets, 131–150
 - for computing answer sets of disjunctive programs, 147–148
 - for computing models of propositional formulas, 131–136
 - resolution, **271**
 - SLD resolution, 276–282
 - SLDNF resolution, 282–284
 - translation from \mathcal{AL} to ASP, 172–174
 - unification, **271–276**
- algorithmic language, **3**
- all_clear()*, 224
- ancestor example, 68–71

- answer set, **11**
 - formal definition, Part I, **21**
 - formal definition, Part II, **26**
 - intuition for, 16
 - of CR-Prolog, **105**
- answer set planning, 197
- answer-set programming paradigm, 114
- Answer Set Prolog (ASP), **5**, 11–35
 - semantics, 16–29
 - syntax, 11–16
- answer set solver, **29**, 114–115
- arithmetic term, 12, 15
- arity, **11**
- ASP program, **14**
 - properties of, 31–35
- ASP solver, *see* answer set solver
- ASPIDE, 310–313
- atom, **13**
 - ground, **14**
- attribute (P-log), **241**
- autoepistemic logic, 44, 46–47
- Awareness Axiom, **225**, 235
- axiomatic method, 6–8

- backtracking, 131, 281, 282
- Bayesian squirrel example, 260–263
- biased dice example, 245–246
- blocks world domain, **154**
 - concurrent actions in, 182–184
 - first try in ASP, 152–162
 - modeling with \mathcal{AL} , 179–182
 - planning, 195–199
- body (of a rule), **14**
- briefcase domain, 175–179
 - recorded history in, 219–220

- Cancellation Axiom, **88**
 - special case, 92
- causal law, **157**, 163, 167, 173
- causal probability, **247**, **247**
- causal probability statement, **245**
- choice rule, **118**, 124, 127, 193
- circuit diagnostic example, 217–218, 221–223, 226–228, 325–329
- circumscription, 44–46
- Clark's completion, **50**, 49–53
- Clark, Keith L., 49
- ClaspD, **115**
- classical planning, **192**
- Clingo, **115**, 307
- Closed World Assumption (CWA), 21, 32, 65–66, 86
- Colmerauer, Alain, 8
- commonsense knowledge, 61, 74, 76, 125, 126
- commonsense reasoning, 20, 86, 236–238
- complementary, **14**
- concurrent planning, 208–209
- configuration (of a system), **220**
- Cons()*, 133
- Cons1()*, 137, 138
- Cons2()*, 144, 145
- consequence
 - ASP, **19**
 - computation for ASP, 138–146
 - computation for FOL, 133–136
 - FOL, 42
 - well-founded, **54**
- consistent
 - set of domain literals, 167
 - set of e-literals, 136
 - set of ground literals, **21**
- constraint, **14**, 34
 - example, 18, 25
 - removing, 35
- Contingency Axiom, **104**, 106
- cowardly students example, 91–92
- CR-Prolog, 103–108, 209–211, 228–229, 263–266
- cr-rule, **104**
- CRModels, 107
- CWA, *see* Closed World Assumption
- Datalog, 8
- Davis-Putnam procedure, 131–136
- declarative language, **4**
- default, **86–108**
 - Contingency Axiom, **104**
 - exception to, 87–108
 - priorities between, 95–98, 102
 - Reiter's, 47
- default logic, 44, 47–49
- default negation, **14**, 49, 52
 - example, 19, 26
 - removal, 26
- default rule, 88
- default theory, **48**
- defined fluent, **162**, 172, 173, 198
- definite program, **140**, 271
- dependency graph, **33**
- derivatives example (Prolog), 295–304
- diagnose()*, 226
- diagnostic problem, **216**
 - example, 217–218
- diagnostics, 216–231
- dice example, 243–245
- DLV, **115**, 307
- domain literal, **166**
 - complete set of, **167**
 - consistent set of, **167**
- domain properties, **166**
- Doyle, Jon, 44
- dynamic domain, 152
 - diagnostics, 216–231
 - modeling, 152–188
 - planning, 192–213
- dynamic range, **242**
 - effect on possible worlds, 250–251
- elaboration tolerance, **5**, 80–81, 154, 184
- electrical circuit example, 71–75
- e-literal, *see* extended literal
- encoding
 - of a system description, **172**
- entailment, *see also* consequence
 - ASP, **19**
 - FOL, 42
 - minimal (circumscription), 44
 - recorded history, **219**
- epistemic disjunction, **14**
 - example, 17–18, 24–25
- Euclid, 7

- exception to a default, 87–108
 - given complete information, 92
 - indirect, **103**–108
 - strong, **88**–93
 - weak, **88**–93
- executability condition, 160, 167, 173
- exogenous action, **216**
- expert knowledge, 61, 74
- expl()*, 226
- explanation, **216**
 - best, **224**
 - computation of, 224–228
 - generation rule, 226
 - minimal, 228–229
 - possible, **221**
- explanation generation rule, 226
- extended literal, **14**, 136
- extension of a default theory, **48**
- fact, **15**
- Fages, François, 53
- family example, 4–6, 62–66
- findall* (Prolog built-in), 290
- first-order logic, 40–43
- fixpoint, **53**
- floundering, 284, 289
- fluent, **155**, 166
 - defined, **162**, 172, 173, 198
 - inertial, **162**, 172
- fluent dependency graph, **170**
- fluent literal, **166**
- FOL, *see* first-order logic
- Frame Problem, **163**
- goal, **192**, 193
- Gringo, **115**
- ground
 - atom, **14**
 - instantiation, **15**
 - literal, **14**
 - term, **12**
- grounding, **15**
- Hamiltonian cycle example, 115–120
- head (of a rule), **14**
- heuristic for ASP solvers, 147
- heuristics for planning, 204–207
- hierarchical information, 75–81, 99–103
- holds()*, 156
- horizon (of a planning problem), **192**
- hpd()*, 219, 229–230
- igniting the burner example, 200–201, 317–320
- inconsistent program, **27**
- indirect exception, **103**
- Inertia Axiom, **159**
 - encoding, 173
 - Frame Problem, 163
- inertial fluent, **162**, 172
- inheritance hierarchy, **76**, 75–81
 - with defaults, 99–103
- interpretation
 - first-order, 41
 - logic programming, 136
 - partial (logic programming), 136
 - partial (propositional), 132
 - propositional, 132
- is_a()*, 78, 100–103
- IsAnswerSet()*, 140
- is_subclass()*, 77, 100–101
- jungle example, 238–240
- knowledge base, 4–6, 14, 61–81
 - with null values, 93–94
- Kowalski, Robert, 8
- l*-polynomial, **297**
- law of the exclusive middle, 14
- LB()*, 138
- Least()*, 141
- Leibniz Dream, 8
- Leibniz, Gottfried Wilhelm, 7
- level mapping, **31**–32
- list (in Prolog), 287–291
- literal, **13**
 - domain, **166**
 - extended, **14**
 - ground, **14**
 - negative, **14**
 - removing, 34
- locally stratified, **32**, 34
- logic program, *see* ASP program
- logic-based approach to agent design, 3–8

- logic-based approach to AI, **8**
- lower bound, 138
- Lparse, 115
- McCarthy, John, 8, 44, 163
- McDermott, Drew, 44
- member()*, 78–79, 100–101
- mgu, *see* most general unifier
- minimal explanation, 228–229
- minimal plan, 209–211
- minimize statement, 211, 229
- missionaries and cannibals example, 201–204, 320–325
- mkatoms*, 308
- mode (of a Prolog predicate), 288
- model
 - FOL, 42
 - mathematical model of an intelligent agent, 2
 - minimal (FOL), 44
 - of a recorded history, **219**
 - probabilistic, 236
 - propositional, 132
 - supported, 51
 - well-founded, 53
- monotonic, **20**
- monotonicity, **20**
- Monty Hall example, 252–255
- Moore, Robert, 44
- most general unifier (mgu), **272**
- mystery puzzle, 125–127
- Narwhal* example, *see* submarine example
- natural language (translation into ASP), 30–31
- negation as failure, *see* default negation
- negation as finite failure, 52, **283**
- negative information, 76, 94
- negative literal, **14**
 - removing, 34
- nlp*, *see* normal logic program
- nondeterminism in \mathcal{AL} , 184–186
- nonmonotonic, **20**, 49, 88
- nonmonotonic logics, 20, 49
- nonmonotonic logics, 43
- normal logic program, **49**, 54, 271
 - tight, **52**
- null value, 93–94
- obs()*, 219, 229–230
- observation, **218**
- occurs()*, 156
- orphan example, 66
 - with incompleteness, 98
- overspecification, 184
- P-log, **238**, 235–270
- parts inventory example (Prolog), 291–295
- Pearl, Judea, 246
- plan, **192**
- planning, 192–213
 - concurrent, 208–209
 - heuristics, 204–207
 - minimal plan, 209–211
- planning problem, **192**
- possible world, **236**
- pr*-atom, **245**
- predicate, 11
- preference relation (CR-Prolog), **104**
- Principle of Indifference, **238**
- probabilistic measure, **237**, 238, 245, 246–247
 - unnormalized, **247**
- probabilistic model, **236**
- probabilistic reasoning, 6, 235–270
- probability function, **237**
- probability of a proposition, **243**
- Prolog, 8, 271–306
 - interpreter, 271–284
 - programming, 284–304
- Przymusiński, Teodor, 56
- query, **20**
 - ASP answer to, **20**
 - SLD resolution answer to, **276**
 - SLDNF resolution answer to, **282**
- Query()*, 148
- query rule, **277**
- Ramification Problem, **164**
- random attribute, **239**
- random selection rule, **239**
- rat example, 255–257
- Rationality Principle, **16**, 19, 35

- Reality Check Axioms, **225**, 230
- recorded history, **218**
 - briefcase domain, 219–220
 - circuit diagnosis example, 222
 - semantics, **219**
 - syntax, **218**
- recursive definition, **70**
- reduct, **26**, 26
- reification, **77**, 76–81, 103, 156n4
- Reiter's default theory, **48**
- Reiter, Raymond, 44
- resolution, 43, **271**, 284
 - SLD, 8, **271**, 276–282
 - SLDNF, 49, 52, 56, **272**, 282–284
 - SLS, 56
- resolvent, **278**
- Ross, Kenneth A., 53
- Roussel, Philippe, 8
- rule (of ASP), **14**

- safety, 64, 315
- Sat()*, 133
- satisfiability (of a rule), **15**
- satisfiability solver, **120**–121, 131–136
- schema, 175
- Schlipf, John S., 53
- signature, **11**
 - \mathcal{AL} , 166, 172
 - FOL, 132
 - P-log, 238
 - sorted, 13
- simple planning module, **193**–194
- SLD derivation, **279**
- SLD resolution, 8, **271**, 276–282
- SLD resolution inference rule, **278**
- SLDNF resolution, 49, 52, 56, **272**, 282–284
- SLS resolution, 56
- Smodels, **115**
- Solver()*, 136
- Solver1()*, 136, 137
- Solver2()*, 136
- sort, **12**
- sorted signature, **12**–13
- SPARC, 65, 108, 314–316
- specificity principle, **102**, 100–103
- spider bite example, 257–260

- stable expansion, **46**
- stable model, 11n1, *see also* answer set
- stable model semantics, 53
 - connection to autoepistemic logic, 47
- state, **169**, 167–171
- state constraint, 158, 167, 173
- static domain, 152
- static literal, **166**
- statics, **155**, 166
- stratification, 32–34
- stratified, **33**, 47, 56
 - locally, **32**, 34
- strong exception, **88**–93
- STUDENT, **5**, 29–30, 310–313
- subclass relation, 76–81, 100–101
- submarine example, 75–81
- substitution, **272**, 276
- Sudoku puzzle, 121–125
- supported model, **51**
- symptom, **216**, **221**
- symptom checking, 225
- system configuration, **220**
- system description, **167**–186

- Tarski, Alfred, 8
- temporal projection, **186**
- term, **12**
 - ground, **12**
- tight normal logic program, **52**
- Touretzky, David S., 102
- transition, **174**, 162–187
- transition diagram, **162**
- transition diagram, 162–187
 - nondeterministic, 184–186
- Tweety example, 100–103

- UB()*, 144
- uncaring John example, 86–91
- unification, **271**, 272–276
- unifier, **272**
- Unique-Name Assumption (UNA), **65**
- unit propagation, 133
- unnormalized probabilistic measure, **247**
- unsafe rule, *see* safety
- upper bound, 144

Van Gelder, Allen, 53
variant, **279**

wandering robot example,
263–266

weak acyclicity, **171**

weak exception, **88–93**

well-founded
consequence, 54
model, 53–**54**
semantics, 53–56
system description, **170**

XSB, 56, 282