

Contents

1	Declarative programming in AnsProlog* : introduction and preliminaries	1
1.1	Motivation: Why AnsProlog*?	3
1.1.1	AnsProlog* vs Prolog	4
1.1.2	AnsProlog* vs Logic programming	4
1.1.3	AnsProlog* vs Default logic	5
1.1.4	AnsProlog* vs Circumscription and classical logic	6
1.1.5	AnsProlog* as a knowledge representation language	6
1.1.6	AnsProlog* implementations: Both a specification and a programming language	7
1.1.7	Applications of AnsProlog*	7
1.2	Answer set frameworks and programs	8
1.2.1	AnsProlog* Programs	11
1.2.2	AnsProlog* notations	14
1.3	Semantics of AnsProlog* programs	16
1.3.1	Answer sets of AnsProlog ^{-not} and AnsProlog ^{-not,⊥} programs	17
1.3.2	Answer sets of AnsProlog and AnsProlog [⊥] programs	21
1.3.3	Answer sets of AnsProlog [¬] and AnsProlog ^{¬,⊥} programs	27
1.3.4	Answer sets of AnsProlog ^{or,⊥} and AnsProlog ^{¬,or,⊥} programs	33
1.3.5	Query entailment	36
1.3.6	A sound approximation : the well-founded semantics	39
1.4	Database queries and AnsProlog* functions	40
1.4.1	Queries and inherent functions	41
1.4.2	Parameters, Values and literal functions	42
1.4.3	The signature functions	43
1.4.4	An AnsProlog* program being functional	44
1.5	Notes and references	44
2	Simple modules for declarative programming with answer sets	46
2.1	Declarative problem solving modules	47
2.1.1	Integrity Constraints	47
2.1.2	Finite enumeration	48
2.1.3	General enumeration but at least one	49
2.1.4	Choice: general enumeration with exactly one	50
2.1.5	Constrained enumeration	51
2.1.6	Propositional satisfiability	52
2.1.7	Closed first-order queries in AnsProlog and AnsProlog [¬]	53
2.1.8	Checking satisfiability of universal quantified boolean formulas (QBFs)	55

2.1.9	Checking satisfiability of existential QBFs	58
2.1.10	Checking satisfiability of Universal-existential QBFs	59
2.1.11	Checking satisfiability of Existential-universal QBFs	62
2.1.12	Smallest, largest, and next in a linear ordering	65
2.1.13	Establishing linear ordering among a set of objects	65
2.1.14	Representing Aggregates	67
2.1.15	Representing classical disjunction conclusions using AnsProlog	69
2.1.16	Representing exclusive-or conclusions using AnsProlog	71
2.1.17	Cardinality Constraints	71
2.1.18	Weight Constraints	72
2.2	Knowledge representation and reasoning modules	73
2.2.1	Normative statements, exceptions, weak exceptions and direct contradictions: the tweety flies story	73
2.2.2	The frame problem and the Yale Turkey shoot	77
2.2.3	Systematic removal of Close World Assumption: an example	80
2.2.4	Reasoning about what is known and what is not	81
2.3	Notes and references	81
3	Principles and properties of declarative programming with answer sets	83
3.1	Basic notions and basic properties	84
3.1.1	Categorical and Coherent programs	84
3.1.2	Relating answer sets and the program rules	84
3.1.3	Conservative extension	86
3.1.4	I/O Specification of a program	87
3.1.5	Compiling AnsProlog programs to classical logic: Clark's completion	90
3.2	Some AnsProlog* subclasses and their basic properties	93
3.2.1	Stratification of AnsProlog Programs	93
3.2.2	Stratification of AnsProlog ^{or} programs	93
3.2.3	Call-consistency	97
3.2.4	Local stratification and perfect model semantics	98
3.2.5	Acyclicity and tightness	99
3.2.6	Atom dependency graph and order-consistency	102
3.2.7	Signing	104
3.2.8	The relation between the AnsProlog subclasses: a summary	105
3.2.9	Head cycle free AnsProlog ^{¬, or} programs	108
3.3	Restricted monotonicity and signed AnsProlog* programs	108
3.3.1	Restricted monotonicity	108
3.3.2	Signed AnsProlog ^{¬, or} programs and their properties.	109
3.4	Analyzing AnsProlog* programs using 'splitting'	113
3.4.1	Splitting sets	114
3.4.2	Application of splitting	116
3.4.3	Splitting sequences	117
3.4.4	Applications of the Splitting sequence theorem	119
3.5	Language independence and language tolerance	120
3.5.1	Adding sorts to answer set frameworks	122
3.5.2	Language Independence	123
3.5.3	Language Tolerance	124

3.5.4	When sorts can be ignored	125
3.6	Interpolating an AnsProlog program	126
3.6.1	The l-functions of AnsProlog and AnsProlog [¬] programs	129
3.6.2	Interpolation of an AnsProlog program and its properties	131
3.6.3	An algorithm for interpolating AnsProlog programs	133
3.6.4	Properties of the transformation \mathcal{I}	136
3.7	Building and refining programs from components: functional specifications and Realization theorems	137
3.7.1	Functional specifications and lp-functions	138
3.7.2	The compositional and refinement operators	139
3.7.3	Realization theorem for incremental extension	141
3.7.4	Realization theorem for interpolation	142
3.7.5	Representing domain completion and realization of input opening . .	143
3.7.6	Realization theorem for input extension	143
3.8	Filter-Abducible AnsProlog ^{¬, or} programs	144
3.8.1	Basic definitions: simple abduction and filtering	145
3.8.2	Abductive reasoning through filtering: semantic conditions	146
3.8.3	Sufficiency conditions for filter-abducibility of AnsProlog ^{¬, or} Programs	150
3.8.4	Necessary conditions for filter-abducibility	151
3.8.5	Weak abductive reasoning vs filtering	152
3.9	Equivalence of programs and semantics preserving transformations	154
3.9.1	Fold/Unfold transformations	154
3.9.2	Replacing disjunctions in the head of rules	157
3.9.3	From AnsProlog to AnsProlog ^{or, -not} and constraints	158
3.9.4	AnsProlog and mixed integer programming	159
3.9.5	Strongly equivalent AnsProlog* programs and the logic of here-and- there	162
3.9.6	Strong equivalence using propositional logic	165
3.9.7	Additional transformations and preservation of strong equivalence .	166
3.10	Notes and references	168
4	Declarative problem solving and reasoning in AnsProlog*	170
4.1	Three well known problem solving tasks	171
4.1.1	N-queens	171
4.1.2	Tile covering of boards with missing squares	176
4.1.3	Who let the Zebra out?	178
4.2	Constraint Satisfaction Problems (CSPs)	183
4.2.1	N-queens as a CSP instance	184
4.2.2	Schur as a CSP instance	185
4.3	Dynamic Constraint Satisfaction Problems (DCSPs)	186
4.3.1	Encoding DCSPs in AnsProlog	186
4.4	Combinatorial graph problems	188
4.4.1	K-colorability	188
4.4.2	Hamiltonian Circuit	188
4.4.3	K-clique	189
4.4.4	Vertex cover	190
4.4.5	Feedback vertex set	191

4.4.6	Kernel	191
4.4.7	Exercise	192
4.5	Prioritized defaults and inheritance hierarchies	192
4.5.1	The language of prioritized defaults	193
4.5.2	The axioms for reasoning with prioritized defaults	193
4.5.3	Modeling inheritance hierarchies using prioritized defaults	196
4.5.4	Exercise	197
4.6	Notes and References	197
5	Reasoning about actions and planning in AnsProlog*	199
5.1	Reasoning in the action description language \mathcal{A}	199
5.1.1	The language \mathcal{A}	201
5.1.2	Temporal projection and its acyclicity in an AnsProlog formulation: π_1	206
5.1.3	Temporal projection in an AnsProlog [⊥] formulation: π_2	209
5.1.4	Temporal projection in AnsProlog [⊥] in presence of incompleteness: π_3	212
5.1.5	Sound reasoning with non-initial observations in AnsProlog [⊥] : π_3 and π_4	215
5.1.6	Assimilating observations using enumeration and constraints: π_5 and π_6	220
5.1.7	Ignoring sorts through language tolerance	223
5.1.8	Filter-abducibility of π_5 and π_6	225
5.1.9	An alternative formulation of temporal projection in AnsProlog: $\pi_{2.nar}$	225
5.1.10	Modifying $\pi_{2.nar}$ for answer set planning: $\pi_{2.planning}$	228
5.2	Reasoning about actions and plan verification in richer domains	229
5.2.1	Allowing executability conditions	229
5.2.2	Allowing static causal propositions	233
5.2.3	Reasoning about parallel execution of actions	238
5.3	Answer set planning examples in extensions of \mathcal{A} and STRIPS	244
5.3.1	A blocks world example in PDDL	245
5.3.2	Simple blocks world in AnsProlog: $\pi_{strips1}(D_{bw}, O_{bw}, G_{bw})$	247
5.3.3	Simple blocks World with domain constraints	251
5.3.4	Adding defined fluents, qualification and ramification to STRIPS	252
5.3.5	Blocks world with conditional effects	256
5.3.6	Navigating a downtown with one-way streets	258
5.3.7	Downtown navigation: planning while driving	260
5.4	Approximate planning when initial state is incomplete	261
5.5	Planning with procedural constraints	262
5.6	Explaining observations through action occurrences and application to diagnosis	269
5.6.1	Specifying and reasoning with histories	270
5.6.2	From reasoning with histories to agents in a dynamic domain	271
5.6.3	Explaining observations	272
5.6.4	Application to diagnosis	273
5.7	Case study: Planning and plan correctness in a Space shuttle reaction control system	274
5.8	Notes and references	277

6	Complexity, expressiveness and other properties of AnsProlog* programs	278
6.1	Complexity and expressiveness	278
6.1.1	The Polynomial hierarchy	279
6.1.2	Polynomial and exponential classes	281
6.1.3	Arithmetical and analytical hierarchy	282
6.1.4	Complexity and expressiveness of languages	283
6.1.5	Technique for proving expressiveness: general forms	287
6.2	Complexity of AnsDatalog* sub-classes	288
6.2.1	Complexity of propositional AnsDatalog ^{-not}	288
6.2.2	Complexity of AnsDatalog ^{-not}	290
6.2.3	Complexity of AnsDatalog	293
6.2.4	Complexity of AnsDatalog ^{or, -not}	297
6.2.5	Complexity of AnsDatalog ^{or}	299
6.2.6	Summary of the complexity results of AnsDatalog* sub-classes	300
6.3	Expressiveness of AnsDatalog* sub-classes	301
6.3.1	Expressiveness of AnsDatalog	302
6.3.2	Expressiveness of AnsDatalog ^{or}	303
6.4	Complexity and expressiveness of AnsProlog* sub-classes	304
6.4.1	Complexity of AnsProlog* sub-classes	305
6.4.2	Summary of complexity results	309
6.4.3	Expressiveness of AnsProlog* sub-classes	309
6.5	Compact representation and compilability of AnsProlog	311
6.6	Relationship with other knowledge representation formalisms	313
6.6.1	Inexistence of modular translations from AnsProlog* to monotonic logics	314
6.6.2	Classical logic and AnsProlog*	314
6.6.3	Circumscription and AnsProlog	316
6.6.4	Autoepistemic logic and AnsProlog*	319
6.6.5	Default logic and AnsProlog*	322
6.6.6	Truth Maintenance Systems and AnsProlog*	325
6.6.7	Description logics and AnsProlog*	326
6.6.8	Answer set entailment as a non-monotonic entailment relation	336
6.6.9	Answer sets as weakest characterizations with certain desirable properties	339
6.7	Notes and references	341
7	Answer set computing algorithms	345
7.1	Branch and bound with WFS: wfs-bb	346
7.1.1	Computing the well-founded semantics	347
7.1.2	The branch and bound algorithm	354
7.1.3	Heuristic for selecting atoms in wfs-bb	357
7.2	The assume-and-reduce algorithm of SLG	358
7.2.1	The main observation	358
7.2.2	The SLG reduction: <i>reduce_{slg}</i>	359
7.2.3	The SLG modification	360
7.2.4	The assume-and-reduce non-deterministic algorithm	362
7.2.5	From assume-and-reduce to SLG	363

7.3	The smodels algorithm	363
7.3.1	The function <i>expand</i> (P, A)	363
7.3.2	The function <i>lookahead</i> (P, A)	368
7.3.3	The function <i>heuristic</i> (P, A)	369
7.3.4	The main function: <i>smodels</i> (P, A)	370
7.3.5	Strategies and tricks for efficient implementation	371
7.4	The dl _v algorithm	372
7.4.1	The function <i>expand_{dlv}</i> (P, I)	374
7.4.2	The function <i>heuristic_{dlv}</i> (P, I)	375
7.4.3	The function <i>isAnswerSet</i> (P, S)	378
7.4.4	The main dl _v function	378
7.4.5	Comparing dl _v with Smodels	379
7.5	Notes and references	379
7.5.1	Other query answering approaches	380
8	Query answering and answer set computing systems	382
8.1	Smodels	382
8.1.1	The ground subset of the input language of lparse	383
8.1.2	Variables and conditional literals and their grounding	388
8.1.3	Other constructs of the lparse language	391
8.1.4	Invoking lparse and smodels	393
8.1.5	Programming in Smodels: Graph colorability	394
8.1.6	Programming in Smodels: Round robin tournament scheduling	395
8.1.7	Programming in Smodels: Mini-ACC tournament scheduling	396
8.1.8	Programming in Smodels: Knapsack problem	401
8.1.9	Programming in Smodels: Single unit combinatorial auction	402
8.2	The dl _v system	403
8.2.1	Some distinguishing features of dl _v	404
8.2.2	Weak constraints in dl _v vs. optimization statements in Smodels	406
8.2.3	Invoking dl _v	407
8.2.4	Single unit combinatorial auction using weak constraints	408
8.2.5	Conformant planning using dl _v	409
8.3	Applications of answer set computing systems	412
8.3.1	Some AnsProlog _{sm} and AnsProlog _{dlv} programming tricks	412
8.3.2	Combinatorial auctions	415
8.3.3	Planning with durative actions and resources using AnsProlog _{sm}	422
8.3.4	Scheduling using AnsProlog _{sm}	428
8.3.5	Specification and verification of active databases using AnsProlog _{sm}	430
8.4	Pure Prolog	440
8.4.1	A unification algorithm and the occur-check step	446
8.4.2	SLDNF and LDNF resolution	447
8.4.3	Sufficiency conditions	450
8.4.4	Examples of applying pure Prolog sufficiency conditions to programs	454
8.5	Notes and references	456

9	Further extensions of and alternatives to AnsProlog*	458
9.1	AnsProlog ^{not, <i>or</i>, \neg, \perp} : allowing not in the head	458
9.2	AnsProlog ^{{not, <i>or</i>, \neg, \perp}*} : allowing nested expressions	461
9.3	AnsProlog ^{\neg, <i>or</i>, <i>K</i>, <i>M</i>} : allowing knowledge and belief operators	466
9.4	Abductive reasoning with AnsProlog: AnsProlog ^{<i>abd</i>}	470
9.5	Domain closure and the universal query problem	471
9.5.1	Parameterized answer sets and \models_{open}	472
9.5.2	Applications of \models_{open}	472
9.6	AnsProlog _{set} : Adding set constructs to AnsProlog	475
9.7	AnsProlog- \prec^{\neg} programs: AnsProlog ^{\neg} programs with ordering	477
9.7.1	The w-answer sets of AnsProlog- \prec^{\neg} programs	478
9.7.2	The d-answer sets of AnsProlog- \prec^{\neg} programs	479
9.7.3	The b-answer sets of AnsProlog- \prec^{\neg} programs	481
9.8	Well-founded semantics of programs with AnsProlog syntax	482
9.8.1	Original characterization using unfounded sets	482
9.8.2	A slightly different iterated fixpoint characterization	483
9.8.3	Alternating fixpoint characterization	484
9.8.4	Stable classes and well-founded semantics	485
9.8.5	Characterizing answer sets and well-founded semantics using argu- mentation	487
9.8.6	3-valued stable models and the well-founded semantics	489
9.9	Well-founded semantics of programs with AnsProlog ^{\neg} syntax	490
9.10	Notes and references	492
	Appendix A: Ordinals, Lattices and fixpoint theory	494
	Ordinals	494
	Fixpoint theory	494
	Transfinite Sequences	495
	Appendix B: Turing machines	496
	Turing Machines	496
	Non-deterministic Turing Machines	497
	Oracle Turing Machines	497
	Bibliography	498
	Index of Notations	519
	Index of Terms	522