Alberto Pettorossi

List of Publications by Year
in descending order

104	1,114	567281 15	552781 26
papers	citations	h-index	g-index

119	119	119	222
all docs	docs citations	times ranked	citing authors

#	Article	IF	Citations
1	Analysis and Transformation of Constrained Horn Clauses for Program Verification. Theory and Practice of Logic Programming, 2022, 22, 974-1042.	1.5	19
2	Removing Algebraic Data Types from Constrained Horn Clauses Using Difference Predicates. Lecture Notes in Computer Science, 2020, , 83-102.	1.3	8
3	Solving Horn Clauses on Inductive Data Types Without Induction â€“ ERRATUM. Theory and Practice of Logic Programming, 2019, 19, 629.	1.5	1
4	Semantics and Controllability of Time-Aware Business Processes*. Fundamenta Informaticae, 2019, 165, 205-244.	0.4	2
5	Property-Based Test Case Generators for Free. Lecture Notes in Computer Science, 2019, , 186-206.	1.3	2
6	Predicate Pairing for program verification. Theory and Practice of Logic Programming, 2018, 18, 126-166.	1.5	9
7	Solving Horn Clauses on Inductive Data Types Without Induction. Theory and Practice of Logic Programming, 2018, 18, 452-469.	1.5	19
8	Predicate Pairing with Abstraction for Relational Verification. Lecture Notes in Computer Science, 2018, , 289-305.	1.3	2
9	Semantics-based generation of verification conditions via program specialization. Science of Computer Programming, 2017, 147, 78-108.	1.9	21
10	Program Verification using Constraint Handling Rules and Array Constraint Generalizations*. Fundamenta Informaticae, 2017, 150, 73-117.	0.4	3
11	Verification of Time-Aware Business Processes Using Constrained Horn Clauses. Lecture Notes in Computer Science, 2017, , 38-55.	1.3	2
12	Verifying Controllability of Time-Aware Business Processes. Lecture Notes in Computer Science, 2017, , 103-118.	1.3	2
13	Relational Verification Through Horn Clause Transformation. Lecture Notes in Computer Science, 2016, , 147-169.	1.3	22
14	Proving correctness of imperative programs by linearizing constrained Horn clauses. Theory and Practice of Logic Programming, 2015, 15, 635-650.	1.5	15
15	A Rule-based Verification Strategy for Array Manipulating Programs. Fundamenta Informaticae, 2015, 140, 329-355.	0.4	9
16	Semantics-based generation of verification conditions by program specialization. , 2015, , .		19
17	Program verification via iterated specialization. Science of Computer Programming, 2014, 95, 149-175.	1.9	30
18	VeriMAP: A Tool for Verifying Programs through Transformations. Lecture Notes in Computer Science, 2014, , 568-574.	1.3	52

#	Article	IF	Citations
19	Verifying Array Programs by Transforming Verification Conditions. Lecture Notes in Computer Science, 2014, , 182-202.	1.3	9
20	Verifying programs via iterated specialization. , 2013, , .		11
21	Controlling Polyvariance for Specialization-based Verification. Fundamenta Informaticae, 2013, 124, 483-502.	0.4	5
22	Proving Theorems by Program Transformation. Fundamenta Informaticae, 2013, 127, 115-134.	0.4	3
23	Generalization strategies for the verification of infinite state systems. Theory and Practice of Logic Programming, 2013, 13, 175-199.	1.5	32
24	Specialization with Constrained Generalization for Software Model Checking. Lecture Notes in Computer Science, 2013, , 51-70.	1.3	4
25	Synthesizing Concurrent Programs Using Answer Set Programming. Fundamenta Informaticae, 2012, 120, 205-229.	0.4	4
26	Improving Reachability Analysis of Infinite State Systems by Specialization. Fundamenta Informaticae, 2012, 119, 281-300.	0.4	8
27	Constraint-based correctness proofs for logic program transformations. Formal Aspects of Computing, 2012, 24, 569-594.	1.8	4
28	Using Real Relaxations during Program Specialization. Lecture Notes in Computer Science, 2012, , 106-122.	1.3	0
29	Program transformation for development, verification, and synthesis of programs. Intelligenza Artificiale, 2011, 5, 119-125.	1.6	4
30	RCRA 2009 Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion. Fundamenta Informaticae, 2011, 107, i-ii.	0.4	0
31	Program Specialization for Verifying Infinite State Systems: An Experimental Evaluation. Lecture Notes in Computer Science, 2011, , 164-183.	1.3	6
32	Improving Reachability Analysis of Infinite State Systems by Specialization. Lecture Notes in Computer Science, 2011, , 165-179.	1.3	3
33	Transformations of logic programs on infinite lists. Theory and Practice of Logic Programming, 2010, 10, 383-399.	1.5	5
34	The Transformational Approach to Program Development. Lecture Notes in Computer Science, 2010, , 112-135.	1.3	3
35	Deciding Full Branching Time Logic by Program Transformation. Lecture Notes in Computer Science, 2010, , 5-21.	1.3	7
36	Advances in Computational Logic (CILC08). Fundamenta Informaticae, 2009, 96, i-ii.	0.4	0

#	Article	IF	Citations
37	A Folding Rule for Eliminating Existential Variables from Constraint Logic Programs. Fundamenta Informaticae, 2009, 96, 373-393.	0.4	0
38	Totally correct logic program transformations viaÂwell-founded annotations. Higher-Order and Symbolic Computation, 2008, 21, 193-234.	0.3	1
39	A Folding Algorithm for Eliminating Existential Variables from Constraint Logic Programs. Lecture Notes in Computer Science, 2008, , 284-300.	1.3	1
40	Derivation of Efficient Logic Programs by Specialization and Reduction of Nondeterminism. , 2008, , 130-177.		0
41	Transformational Verification of Parameterized Protocols Using Array Formulas. Lecture Notes in Computer Science, 2006, , 23-43.	1.3	1
42	Proving Properties of Constraint Logic Programs by Eliminating Existential Variables. Lecture Notes in Computer Science, 2006, , 179-195.	1.3	5
43	Derivation of Efficient Logic Programs by Specialization and Reduction of Nondeterminism. Higher-Order and Symbolic Computation, 2005, 18, 121-210.	0.3	2
44	Automatic Proofs of Protocols via Program Transformation. , 2005, , 99-116.		0
45	A theory of totally correct logic program transformations. , 2004, , .		3
46	Transformations of logic programs with goals as arguments. Theory and Practice of Logic Programming, 2004, 4, 495-537.	1.5	4
47	Combining Logic Programs and Monadic Second Order Logics by Program Transformation. Lecture Notes in Computer Science, 2003, , 160-181.	1.3	1
48	The List Introduction Strategy for the Derivation of Logic Programs. Formal Aspects of Computing, 2002, 13, 233-251.	1.8	4
49	Verification of Sets of Infinite State Processes Using Program Transformation. Lecture Notes in Computer Science, 2002, , 111-128.	1.3	10
50	Program Derivation = Rules + Strategies. Lecture Notes in Computer Science, 2002, , 273-309.	1.3	7
51	Automated Strategies for Specializing Constraint Logic Programs. Lecture Notes in Computer Science, 2001, , 125-146.	1.3	17
52	Rules and Strategies for Contextual Specialization of Constraint Logic Programs. Electronic Notes in Theoretical Computer Science, 2000, 30, 129-144.	0.9	4
53	Transformation Rules for Logic Programs with Goals as Arguments. Lecture Notes in Computer Science, 2000, , 176-195.	1.3	1
54	Synthesis and transformation of logic programs using unfold/fold proofs. The Journal of Logic Programming, 1999, 41, 197-230.	1.7	36

#	Article	IF	Citations
55	Program specialization via algorithmic unfold/fold transformations. ACM Computing Surveys, 1998, 30, 6.	23.0	1
56	Reducing nondeterminism while specializing logic programs. , 1997, , .		16
57	Enhancing partial deduction via unfold/fold rules. Lecture Notes in Computer Science, 1997, , 146-168.	1.3	3
58	Future directions in program transformation. ACM SIGPLAN Notices, 1997, 32, 99-102.	0.2	1
59	Program Derivation via List Introduction. IFIP Advances in Information and Communication Technology, 1997, , 296-323.	0.7	5
60	Developing correct and efficient logic programs by transformation. Knowledge Engineering Review, 1996, 11, 347-360.	2.6	2
61	Rules and strategies for transforming functional and logic programs. ACM Computing Surveys, 1996, 28, 360-414.	23.0	108
62	A comparative revisitation of some program transformation techniques. Lecture Notes in Computer Science, 1996, , 355-385.	1.3	15
63	A theory of logic program specialization and generalization for dealing with input data properties. Lecture Notes in Computer Science, 1996, , 386-408.	1.3	6
64	Future directions in program transformation. ACM Computing Surveys, 1996, 28, 171.	23.0	5
65	Unfolding-definition-folding, in this order, for avoiding unnecessary variables in logic programs. Theoretical Computer Science, 1995, 142, 89-124.	0.9	47
66	Transformation of logic programs: Foundations and techniques. The Journal of Logic Programming, 1994, 19-20, 261-320.	1.7	143
67	Synthesis of Programs from Unfold/Fold Proofs. Workshops in Computing, 1994, , 141-158.	0.4	6
68	The loop absorption and the generalization strategies for the development of logic programs and partial deduction. The Journal of Logic Programming, 1993, 16, 123-161.	1.7	29
69	Rules and strategies for program transformation. Lecture Notes in Computer Science, 1993, , 263-304.	1.3	7
70	Best-first Strategies for Incremental Transformations of Logic Programs. Workshops in Computing, 1993, , 82-98.	0.4	2
71	An Abstract Strategy for Transforming Logic Programs 1. Fundamenta Informaticae, 1993, 18, 267-286.	0.4	3
72	The Use of the Tupling Strategy in the Development of Parallel Programs. , 1993, , 111-151.		1

#	Article	IF	Citations
73	Semantics preserving transformation rules for Prolog. ACM SIGPLAN Notices, 1991, 26, 274-284.	0.2	12
74	Semantics preserving transformation rules for Prolog. , 1991, , .		15
75	Unfolding â€" definition â€" folding, in this order, for avoiding unnecessary variables in logic programs. Lecture Notes in Computer Science, 1991, , 347-358.	1.3	23
76	Synthesis of eureka predicates for developing logic programs. Lecture Notes in Computer Science, 1990, , 306-325.	1.3	20
77	Observers, experiments, and agents: A comprehensive approach to parallelism. Lecture Notes in Computer Science, 1990, , 375-406.	1.3	6
78	DERIVATION OF PROGRAMS WHICH TRAVERSE THEIR INPUT DATA ONLY ONCE. , 1989, , 165-184.		4
79	Higher order generalization in program derivation. , 1987, , 182-196.		8
80	Derivation of efficient programs for computing sequences of actions. Theoretical Computer Science, 1987, 53, 151-167.	0.9	4
81	Enriched categories for local and interaction calculi. Lecture Notes in Computer Science, 1987, , 57-70.	1.3	4
82	Program development using lambda abstraction. Lecture Notes in Computer Science, 1987, , 420-434.	1.3	9
83	Transformation strategies for deriving on line programs. , 1986, , 127-141.		1
84	Categorical models of process cooperation. Lecture Notes in Computer Science, 1986, , 282-298.	1.3	3
85	Towers of Hanoi problems: Deriving iterative solutions by program transformations. BIT Numerical Mathematics, 1985, 25, 327-334.	2.0	7
86	A methodology for improving parallel programs by adding communications. Lecture Notes in Computer Science, 1985, , 228-250.	1.3	1
87	Higher-order communications for concurrent programming. Parallel Computing, 1984, 1, 331-336.	2.1	0
88	A powerful strategy for deriving efficient programs by transformation. , 1984, , .		45
89	Towards a theory of parallelism and communications for increasing efficiency in applicative languages. Lecture Notes in Computer Science, 1983, , 224-249.	1.3	0
90	Deriving very efficient algorithms for evaluating linear recurrence relations using the program transformation technique. Acta Informatica, 1982, 18, 181.	0.5	20

#	Article	IF	Citations
91	A property which guarantees termination in weak combinatory logic and subtree replacement systems.. Notre Dame Journal of Formal Logic, 1981, 22, 344.	0.4	0
92	A transformational approach for developing parallel programs. Lecture Notes in Computer Science, 1981, , 245-258.	1.3	1
93	Comparing and putting together recursive path ordering, simplification orderings and Non-Ascending Property for termination proofs of term rewriting systems. Lecture Notes in Computer Science, 1981, , 432-447.	1.3	4
94	An approach to communications and parallelism in applicative languages. Lecture Notes in Computer Science, 1981, , 432-446.	1.3	0
95	Derivation of an O(k2 log n) algorithm for computing order-k fibonacci numbers from the O(k3log n) matrix multiplication method. Information Processing Letters, 1980, 11, 172-179.	0.6	10
96	On subrecursiveness in weak combinatory logic. Lecture Notes in Computer Science, 1975, , 297-311.	1.3	4
97	Verification of Imperative Programs by Constraint Logic Program Transformation. Electronic Proceedings in Theoretical Computer Science, EPTCS, 0, 129, 186-210.	0.8	6
98	Removing Unnecessary Variables from Horn Clause Verification Conditions. Electronic Proceedings in Theoretical Computer Science, EPTCS, 0, 219, 49-55.	0.8	0
99	Bounded Symbolic Execution for Runtime Error Detection of Erlang Programs. Electronic Proceedings in Theoretical Computer Science, EPTCS, 0, 278, 19-26.	0.8	1
100	Proving Properties of Sorting Programs: A Case Study in Horn Clause Verification. Electronic Proceedings in Theoretical Computer Science, EPTCS, 0, 296, 48-75.	0.8	2
101	Lemma Generation for Horn Clause Satisfiability: A Preliminary Study. Electronic Proceedings in Theoretical Computer Science, EPTCS, 0, 299, 4-18.	0.8	3
102	A Historical Account of My Early Research Interests. Electronic Proceedings in Theoretical Computer Science, EPTCS, 0, 320, 1-28.	0.8	0
103	Satisfiability of constrained Horn clauses on algebraic data types: A transformation-based approach. Journal of Logic and Computation, 0, , .	0.8	3
104	Verifying Catamorphism-Based Contracts using Constrained Horn Clauses. Theory and Practice of Logic Programming, 0, , 1-18.	1.5	6