

Maurizio Proietti

List of Publications by Year in descending order

Source: <https://exaly.com/author-pdf/9017304/publications.pdf>

Version: 2024-02-01

74
papers

918
citations

623734

14
h-index

552781

26
g-index

81
all docs

81
docs citations

81
times ranked

220
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 – 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	Program Verification using Constraint Handling Rules and Array Constraint Generalizations*. Fundamenta Informaticae, 2017, 150, 73-117.	0.4	3
10	Verification of Time-Aware Business Processes Using Constrained Horn Clauses. Lecture Notes in Computer Science, 2017, , 38-55.	1.3	2
11	Verifying Controllability of Time-Aware Business Processes. Lecture Notes in Computer Science, 2017, , 103-118.	1.3	2
12	Relational Verification Through Horn Clause Transformation. Lecture Notes in Computer Science, 2016, , 147-169.	1.3	22
13	Proving correctness of imperative programs by linearizing constrained Horn clauses. Theory and Practice of Logic Programming, 2015, 15, 635-650.	1.5	15
14	A Rule-based Verification Strategy for Array Manipulating Programs. Fundamenta Informaticae, 2015, 140, 329-355.	0.4	9
15	VeriMAP: A Tool for Verifying Programs through Transformations. Lecture Notes in Computer Science, 2014, , 568-574.	1.3	52
16	Behavioral Reasoning on Semantic Business Processes in a Rule-Based Framework. Communications in Computer and Information Science, 2014, , 293-313.	0.5	6
17	Verifying Array Programs by Transforming Verification Conditions. Lecture Notes in Computer Science, 2014, , 182-202.	1.3	9
18	Verifying programs via iterated specialization. , 2013, , .		11

#	ARTICLE	IF	CITATIONS
19	Controlling Polyvariance for Specialization-based Verification. <i>Fundamenta Informaticae</i> , 2013, 124, 483-502.	0.4	5
20	Proving Theorems by Program Transformation. <i>Fundamenta Informaticae</i> , 2013, 127, 115-134.	0.4	3
21	Generalization strategies for the verification of infinite state systems. <i>Theory and Practice of Logic Programming</i> , 2013, 13, 175-199.	1.5	32
22	A Semantic Framework for Knowledge Management in Virtual Innovation Factories. <i>International Journal of Information System Modeling and Design</i> , 2013, 4, 70-92.	1.1	14
23	Specialization with Constrained Generalization for Software Model Checking. <i>Lecture Notes in Computer Science</i> , 2013, , 51-70.	1.3	4
24	Synthesizing Concurrent Programs Using Answer Set Programming. <i>Fundamenta Informaticae</i> , 2012, 120, 205-229.	0.4	4
25	Improving Reachability Analysis of Infinite State Systems by Specialization. <i>Fundamenta Informaticae</i> , 2012, 119, 281-300.	0.4	8
26	Constraint-based correctness proofs for logic program transformations. <i>Formal Aspects of Computing</i> , 2012, 24, 569-594.	1.8	4
27	Ontology-Based Querying of Composite Services. <i>Lecture Notes in Computer Science</i> , 2012, , 159-180.	1.3	17
28	Using Real Relaxations during Program Specialization. <i>Lecture Notes in Computer Science</i> , 2012, , 106-122.	1.3	0
29	Querying Semantically Enriched Business Processes. <i>Lecture Notes in Computer Science</i> , 2011, , 294-302.	1.3	10
30	Program Specialization for Verifying Infinite State Systems: An Experimental Evaluation. <i>Lecture Notes in Computer Science</i> , 2011, , 164-183.	1.3	6
31	Improving Reachability Analysis of Infinite State Systems by Specialization. <i>Lecture Notes in Computer Science</i> , 2011, , 165-179.	1.3	3
32	Transformations of logic programs on infinite lists. <i>Theory and Practice of Logic Programming</i> , 2010, 10, 383-399.	1.5	5
33	The Transformational Approach to Program Development. <i>Lecture Notes in Computer Science</i> , 2010, , 112-135.	1.3	3
34	Deciding Full Branching Time Logic by Program Transformation. <i>Lecture Notes in Computer Science</i> , 2010, , 5-21.	1.3	7
35	A Folding Rule for Eliminating Existential Variables from Constraint Logic Programs. <i>Fundamenta Informaticae</i> , 2009, 96, 373-393.	0.4	0
36	Totally correct logic program transformations via well-founded annotations. <i>Higher-Order and Symbolic Computation</i> , 2008, 21, 193-234.	0.3	1

#	ARTICLE	IF	CITATIONS
37	A Folding Algorithm for Eliminating Existential Variables from Constraint Logic Programs. Lecture Notes in Computer Science, 2008, , 284-300.	1.3	1
38	Derivation of Efficient Logic Programs by Specialization and Reduction of Nondeterminism. , 2008, , 130-177.		0
39	Transformational Verification of Parameterized Protocols Using Array Formulas. Lecture Notes in Computer Science, 2006, , 23-43.	1.3	1
40	Proving Properties of Constraint Logic Programs by Eliminating Existential Variables. Lecture Notes in Computer Science, 2006, , 179-195.	1.3	5
41	Derivation of Efficient Logic Programs by Specialization and Reduction of Nondeterminism. Higher-Order and Symbolic Computation, 2005, 18, 121-210.	0.3	2
42	Transformations of logic programs with goals as arguments. Theory and Practice of Logic Programming, 2004, 4, 495-537.	1.5	4
43	The List Introduction Strategy for the Derivation of Logic Programs. Formal Aspects of Computing, 2002, 13, 233-251.	1.8	4
44	Verification of Sets of Infinite State Processes Using Program Transformation. Lecture Notes in Computer Science, 2002, , 111-128.	1.3	10
45	Program Derivation = Rules + Strategies. Lecture Notes in Computer Science, 2002, , 273-309.	1.3	7
46	Automated Strategies for Specializing Constraint Logic Programs. Lecture Notes in Computer Science, 2001, , 125-146.	1.3	17
47	Rules and Strategies for Contextual Specialization of Constraint Logic Programs. Electronic Notes in Theoretical Computer Science, 2000, 30, 129-144.	0.9	4
48	Transformation Rules for Logic Programs with Goals as Arguments. Lecture Notes in Computer Science, 2000, , 176-195.	1.3	1
49	Synthesis and transformation of logic programs using unfold/fold proofs. The Journal of Logic Programming, 1999, 41, 197-230.	1.7	36
50	Program specialization via algorithmic unfold/fold transformations. ACM Computing Surveys, 1998, 30, 6.	23.0	1
51	Reducing nondeterminism while specializing logic programs. , 1997, , .		16
52	Enhancing partial deduction via unfold/fold rules. Lecture Notes in Computer Science, 1997, , 146-168.	1.3	3
53	Program Derivation via List Introduction. IFIP Advances in Information and Communication Technology, 1997, , 296-323.	0.7	5
54	Developing correct and efficient logic programs by transformation. Knowledge Engineering Review, 1996, 11, 347-360.	2.6	2

#	ARTICLE	IF	CITATIONS
55	Rules and strategies for transforming functional and logic programs. ACM Computing Surveys, 1996, 28, 360-414.	23.0	108
56	A comparative revisitaton of some program transformation techniques. Lecture Notes in Computer Science, 1996, , 355-385.	1.3	15
57	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
58	Future directions in program transformation. ACM Computing Surveys, 1996, 28, 171.	23.0	5
59	Unfolding-definition-folding, in this order, for avoiding unnecessary variables in logic programs. Theoretical Computer Science, 1995, 142, 89-124.	0.9	47
60	Transformation of logic programs: Foundations and techniques. The Journal of Logic Programming, 1994, 19-20, 261-320.	1.7	143
61	Synthesis of Programs from Unfold/Fold Proofs. Workshops in Computing, 1994, , 141-158.	0.4	6
62	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
63	Semantics preserving transformation rules for Prolog. ACM SIGPLAN Notices, 1991, 26, 274-284.	0.2	12
64	Semantics preserving transformation rules for Prolog. , 1991, , .		15
65	Unfolding " definition " folding, in this order, for avoiding unnecessary variables in logic programs. Lecture Notes in Computer Science, 1991, , 347-358.	1.3	23
66	Verification of Programs by Combining Iterated Specialization with Interpolation. Electronic Proceedings in Theoretical Computer Science, EPTCS, 0, 169, 3-18.	0.8	2
67	Verification of Imperative Programs by Constraint Logic Program Transformation. Electronic Proceedings in Theoretical Computer Science, EPTCS, 0, 129, 186-210.	0.8	6
68	Removing Unnecessary Variables from Horn Clause Verification Conditions. Electronic Proceedings in Theoretical Computer Science, EPTCS, 0, 219, 49-55.	0.8	0
69	Bounded Symbolic Execution for Runtime Error Detection of Erlang Programs. Electronic Proceedings in Theoretical Computer Science, EPTCS, 0, 278, 19-26.	0.8	1
70	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
71	Lemma Generation for Horn Clause Satisfiability: A Preliminary Study. Electronic Proceedings in Theoretical Computer Science, EPTCS, 0, 299, 4-18.	0.8	3
72	Transformational Verification of Quicksort. Electronic Proceedings in Theoretical Computer Science, EPTCS, 0, 320, 95-109.	0.8	0

#	ARTICLE	IF	CITATIONS
73	Satisfiability of constrained Horn clauses on algebraic data types: A transformation-based approach. Journal of Logic and Computation, 0, , .	0.8	3
74	Verifying Catamorphism-Based Contracts using Constrained Horn Clauses. Theory and Practice of Logic Programming, 0, , 1-18.	1.5	6