Andrey Rybalchenko

List of Publications by Year
in descending order

30 papers	2,047 citations	471509 17 h-index	580821 25 g-index
31 all docs	31 docs citations	31 times ranked	761 citing authors

#	Article	IF	Citations
1	Proving program termination. Communications of the ACM, 2011, 54, 88-98.	4.5	322
2	A Complete Method for the Synthesis of Linear Ranking Functions. Lecture Notes in Computer Science, 2004, , 239-251.	1.3	273
3	Termination proofs for systems code. , 2006, , .		212
4	Synthesizing software verifiers from proof rules. , 2012, , .		138
5	ARMC: The Logical Choice for Software Model Checking with Abstraction Refinement. Lecture Notes in Computer Science, 2006, , 245-259.	1.3	96
6	InvGen: An Efficient Invariant Generator. Lecture Notes in Computer Science, 2009, , 634-640.	1.3	92
7	Termination proofs for systems code. ACM SIGPLAN Notices, 2006, 41, 415-426.	0.2	89
8	Proving non-termination. , 2008, , .		89
9	Abstraction Refinement for Termination. Lecture Notes in Computer Science, 2005, , 87-101.	1.3	78
10	Synthesizing software verifiers from proof rules. ACM SIGPLAN Notices, 2012, 47, 405-416.	0.2	65
11	Invariant Synthesis for Combined Theories. Lecture Notes in Computer Science, 2007, , 378-394.	1.3	64
12	Proving that programs eventually do something good. , 2007, , .		63
13	Proving Conditional Termination. Lecture Notes in Computer Science, 2008, , 328-340.	1.3	53
14	Constraint Solving for Interpolation. , 2007, , 346-362.		50
15	Transition predicate abstraction and fair termination. , 2005, , .		50
16	From Tests to Proofs. Lecture Notes in Computer Science, 2009, , 262-276.	1.3	44
17	Model checking Duration Calculus: a practical approach. Formal Aspects of Computing, 2008, 20, 481-505.	1.8	29
18	Path invariants. ACM SIGPLAN Notices, 2007, 42, 300-309.	0.2	25

#	Article	IF	Citations
19	Summarization for termination: no return!. Formal Methods in System Design, 2009, 35, 369-387.	0.8	21
20	Proving that programs eventually do something good. ACM SIGPLAN Notices, 2007, 42, 265-276.	0.2	20
21	Heap Assumptions on Demand. Lecture Notes in Computer Science, 2008, , 314-327.	1.3	19
22	Separating Fairness and Well-Foundedness for the Analysis of Fair Discrete Systems. Lecture Notes in Computer Science, 2005, , 124-139.	1.3	17
23	Applying Prolog to develop distributed systems. Theory and Practice of Logic Programming, 2010, 10, 691-707.	1.5	14
24	Distributed and Predictable Software Model Checking. Lecture Notes in Computer Science, 2011, , 340-355.	1.3	11
25	Transition predicate abstraction and fair termination. ACM SIGPLAN Notices, 2005, 40, 132-144.	0.2	10
26	From tests to proofs. International Journal on Software Tools for Technology Transfer, 2013, 15, 291-303.	1.9	8
27	Cardinality Abstraction for Declarative Networking Applications. Lecture Notes in Computer Science, 2009, , 584-598.	1.3	8
28	Binary Reachability Analysis of Higher Order Functional Programs. Lecture Notes in Computer Science, 2012, , 388-404.	1.3	5
29	Recursive Games for Compositional Program Synthesis. Lecture Notes in Computer Science, 2016, , 19-39.	1.3	2
30	Preface: Special Issue on Interpolation. Journal of Automated Reasoning, 2016, 57, 1-2.	1.4	1