Analysis of Types of Self-Improving Software

Roman V. Yampolskiy(✉)

Computer Engineering and Computer Science, Speed School of Engineering,
University of Louisville, Louisville, USA
roman.yampolskiy@louisville.edu

Abstract. Software capable of improving itself has been a dream of computer scientists since the inception of the field. In this work we provide definitions for Recursively Self-Improving software, survey different types of self-improving software, and provide a review of the relevant literature. Finally, we address security implications from self-improving intelligent software.

Keywords: Recursive self-improvement · Self-modifying code · Self-modifying software · Self-modifying algorithm · Autogenous intelligence · Bootstrap fallacy

1 Introduction

Since the early days of computer science, visionaries in the field anticipated creation of a self-improving intelligent system, frequently as an easier pathway to creation of true artificial intelligence[1]. As early as 1950 Alan Turing wrote: "Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain. Presumably the child-brain is something like a notebook as one buys from the stationers. Rather little mechanism, and lots of blank sheets... Our hope is that there is so little mechanism in the child-brain that something like it can be easily programmed. The amount of work in the education we can assume, as a first approximation, to be much the same as for the human child" [1].

Turing's approach to creation of artificial (super)intelligence was echoed by I.J. Good, Marvin Minsky and John von Neumann, all three of whom published on it (interestingly in the same year, 1966): Good - "Let an ultraintelligent machine be defined as a machine that can far surpass all the intellectual activities of any man however clever. Since the design of machines is one of these intellectual activities, an ultraintelligent machine could design even better machines; there would then unquestionably be an 'intelligence explosion,' and the intelligence of man would be left far behind. Thus the first ultraintelligent machine is the last invention that man need ever make" [2]. Minsky - "Once we have devised programs with a genuine capacity for self-improvement a rapid evolutionary process will begin. As the machine improves both itself and its model of itself, we shall begin to see all the phenomena associated

[1]This paper is based on material excerpted, with permission, from the book - Artificial Superintelligence: a Futuristic Approach © 2015 CRC Press.

J. Bieger (Ed.): AGI 2015, LNAI 9205, pp. 384–393, 2015.
DOI: 10.1007/978-3-319-21365-1_39

with the terms "consciousness," "intuition" and "intelligence" itself. It is hard to say how close we are to this threshold, but once it is crossed the world will not be the same" [3]. Von Neumann - "There is thus this completely decisive property of complexity, that there exists a critical size below which the process of synthesis is degenerative, but above which the phenomenon of synthesis, if properly arranged, can become explosive, in other words, where syntheses of automata can proceed in such a manner that each automaton will produce other automata which are more complex and of higher potentialities than itself" [4]. Similar types of arguments are still being made today by modern researchers and the area of RSI research continues to grow in popularity [5-7], though some [8] have argued that recursive self-improvement process requires hyperhuman capability to "get the ball rolling", a kind of "Catch 22"

2 Taxonomy of Types of Self-Improvement

Self-improving software can be classified by the degree of self-modification it entails. In general we distinguish three levels of improvement – modification, improvement (weak self-improvement) and recursive improvement (strong self-improvement). However, it is easy to see that recursive improvement is a type/subset of improvement which is a subset of modification. For the purposes of this paper we will treat them as separate classes.

2.1 Self-Modification

Self-Modification does not produce improvement and is typically employed for code obfuscation to protect software from being reverse engineered or to disguise self-replicating computer viruses from detection software. While a number of obfuscation techniques are known to exist [9], ex. self-modifying code [10], polymorphic code, metamorphic code, diversion code [11], none of them are intended to modify the underlying algorithm. The sole purpose of such approaches is to modify how the source code looks to those trying to understand the software in questions and what it does [12].

2.2 Self-Improvement

Self-Improvement or Self-adaptation [13] is a desirable property of many types of software products [14] and typically allows for some optimization or customization of the product to the environment and users it is deployed with. Common examples of such software include evolutionary algorithms such as Genetic Algorithms [15-20] or Genetic Programming which optimize software parameters with respect to some well understood fitness function and perhaps work over some highly modular programming language to assure that all modifications result in software which can be compiled and evaluated. The system may try to optimize its components by creating internal tournaments between candidate solutions. Omohundro proposed the concept of efficiency drives in self-improving software [21]. Because of one of such drives, balance drive, self-improving systems will tend to balance the allocation of resources between their different subsystems. If the system is not balanced overall performance

of the system could be increased by shifting resources from subsystems with small marginal improvement to those with larger marginal increase [21]. While performance of the software as a result of such optimization may be improved the overall algorithm is unlikely to be modified to a fundamentally more capable one.

Additionally, the law of diminishing returns quickly sets in and after an initial significant improvement phase, characterized by discovery of "low-hanging fruit", future improvements are likely to be less frequent and less significant, producing a Bell curve of valuable changes. Metareasoning, metalearning, learning to learn, and lifelong learning are terms which are often used in the machine learning literature to indicate self-modifying learning algorithms or the process of selecting an algorithm which will perform best in a particular problem domain [22]. Yudkowsky calls such process *non-recursive optimization* – a situation in which one component of the system does the optimization and another component is getting optimized [23].

In the field of complex dynamic systems, aka chaos theory, positive feedback systems are well known to always end up in what is known as an *attractor*- a region within system's state space that the system can't escape from [24]. A good example of such attractor convergence is the process of Metacompilation or Supercompilation [25] in which a program designed to take source code written by a human programmer and to optimize it for speed is applied to its own source code. It will likely produce a more efficient compiler on the first application perhaps by 20%, on the second application by 3%, and after a few more recursive iterations converge to a fixed point of zero improvement [24].

2.3 Recursive Self-Improvement

Recursive Self-Improvement is the only type of improvement which has potential to completely replace the original algorithm with a completely different approach and more importantly to do so multiple times. At each stage newly created software should be better at optimizing future version of the software compared to the original algorithm. As of the time of this writing it is a purely theoretical concept with no working RSI software known to exist. However, as many have predicted that such software might become a reality in the 21st century it is important to provide some analysis of properties such software would exhibit.

Self-modifying and self-improving software systems are already well understood and are quite common. Consequently, we will concentrate exclusively on RSI systems. In practice performance of almost any system can be trivially improved by allocation of additional computational resources such as more memory, higher sensor resolution, faster processor or greater network bandwidth for access to information. This linear scaling doesn't fit the definition of recursive-improvement as the system doesn't become better at improving itself. To fit the definition the system would have to engineer a faster type of memory not just purchase more memory units of the type it already has access to. In general hardware improvements are likely to speed up the system, while software improvements (novel algorithms) are necessary for achievement of meta-improvements.

It is believed that AI systems will have a number of advantages over human programmers making it possible for them to succeed where we have so far failed. Such advantages include [26]: longer work spans (no breaks, sleep, vocation, etc.), omniscience (expert level knowledge in all fields of science, absorbed knowledge of all published works), superior computational resources (brain vs processor, human memory vs RAM), communication speed (neurons vs wires), increased serial depth (ability to perform sequential operations in access of about a 100 human brain can manage), duplicability (intelligent software can be instantaneously copied), editability (source code unlike DNA can be quickly modified), goal coordination (AI copies can work towards a common goal without much overhead), improved rationality (AIs are likely to be free from human cognitive biases) [27], new sensory modalities (native sensory hardware for source code), blending over of deliberative and automatic processes (management of computational resources over multiple tasks), introspective perception and manipulation (ability to analyze low level hardware, ex. individual neurons), addition of hardware (ability to add new memory, sensors, etc.), advanced communication (ability to share underlying cognitive representations for memories and skills) [28].

Chalmers [29] uses logic and mathematical induction to show that if an AI_0 system is capable of producing only slightly more capable AI_1 system generalization of that process leads to superintelligent performance in AI_n after n generations. He articulates, that his proof assumes that the *proportionality thesis,* which states that increases in intelligence lead to proportionate increases in the capacity to design future generations of AIs, is true.

Nivel et al. proposed formalization of RSI systems as autocatalytic sets – collections of entities comprised of elements, each of which can be created by other elements in the set making it possible for the set to self-maintain and update itself. They also list properties of a system which make it purposeful, goal-oriented and self-organizing, particularly: *reflectivity* – ability to analyze and rewrite its own structure; *autonomy* – being free from influence by system's original designers (*bounded autonomy* – is a property of a system with elements which are not subject to self-modification); *endogeny* – an autocatalytic ability [30]. Nivel and Thorisson also attempt to operationalize autonomy by the concept of *self-programming* which they insist has to be done in an experimental way instead of a theoretical way (via proofs of correctness) since it is the only tractable approach [31].

Yudkowsky writes prolifically about recursive self-improving processes and suggests that introduction of certain concepts might be beneficial to the discussion, specifically he proposes use of terms - Cascades, Cycles and Insight which he defines as: Cascades – when one development leads to another; Cycles – repeatable cascade in which one optimization leads to another which in turn benefits the original optimization; Insight – new information which greatly increases one's optimization ability [32]. Yudkowsky also suggests that the goodness and number of opportunities in the space of solutions be known as *Optimization Slope* while *optimization resources* and *optimization efficiency* refer to how much of computational resources an agent has access to and how efficiently the agent utilizes said resources. An agent engaging in an *optimization process* and able to hit non-trivial targets in large search space [33] is described as having significant optimization power [23].

3 RSI Software Classification

RSI software could be classified based on the number of improvements it is capable of achieving. The most trivial case is the system capable of undergoing a single fundamental improvement. The hope is that truly RSI software will be capable of many such improvements, but the question remains open regarding the possibility of an infinite number of recursive-improvements. It is possible that some upper bound on improvements exists limiting any RSI software to a finite number of desirable and significant rewrites. Critics explain failure of scientists, to date, to achieve a sustained RSI process by saying that RSI researchers have fallen victims of the *bootstrap fallacy* [34].

3.1 How Improvements are Discovered

Another axis on which RSI systems can be classified has to do with how improvements are discovered. Two fundamentally different approaches are understood to exist. The first one is a brute force based approach [35] which utilizes Levin (Universal [36]) Search [37]. The idea is to consider all possible strings of source code up to some size limit and to select the one which can be proven to provide improvements. While theoretically optimal and guaranteed to find superior solution if one exists this method is not computationally feasible in practice. Some variants of this approach to self-improvement, known as Gödel Machines [38-43], Optimal Ordered Problem Solver (OOPS) [44] and Incremental Self-Improvers [45, 46], have been thoroughly analyzed by Schmidhuber and his co-authors. Second approach assumes that the system has a certain level of scientific competence and uses it to engineer and test its own replacement. Whether a system of any capability can intentionally invent a more capable and so a more complex system remains as the fundamental open problem of RSI research.

It is important to note that the first concrete algorithms for RSI were all by Schmidhuber. His diploma thesis from 1987 already was about an evolutionary system that learns to inspect and improve its own learning algorithm, where Genetic Programming (GP) is applied to itself, to recursively evolve better GP methods. His RSI based on the self-referential Success-Story Algorithm for self-modifying probabilistic programs was already able to solve complex tasks [47]. And finally, his self-referential recurrent neural networks run and inspect and change their own weight change algorithms [48]. In 2001, his former student Hochreiter had actually a practical implementation of such an RNN that learns an excellent learning algorithm, at least for the limited domain of quadratic functions [49, 50].

3.2 Hybrid Systems

Finally, we can consider a hybrid RSI system which includes both an artificially intelligent program and a human scientist. Mixed human-AI teams have been very successful in many domains such as chess or theorem proving. It would be surprising if having a combination of natural and artificial intelligence did not provide an advantage in designing new AI systems or enhancing biological intelligence. We are currently experiencing a limited version of this approach with human computer scientists developing

progressively better versions of AI software (while utilizing continuously improving software tools), but since the scientists themselves remain unenhanced we can't really talk about self-improvement. This type of RSI can be classified as Indirect recursive improvement as opposed to Direct RSI in which the system itself is responsible for all modifications. Other types of Indirect RSI may be based on collaboration between multiple artificial systems instead of AI and human teams [51].

3.3 Other Properties

In addition to classification with respect to types of RSI we can also evaluate systems as to certain binary properties. For example: We may be interested only in systems which are guaranteed not to decrease in intelligence, even temporarily, during the improvement process. This may not be possible if the intelligence design landscape contains local maxima points.

Another property of any RSI system we are interested in understanding better is necessity of unchanging source code segments. In other words must an RSI system be able to modify any part of its source code or are certain portions of the system (encoded goals, verification module) must remain unchanged from generation to generation. Such portions would be akin to ultra-conserved elements or conserved sequences of DNA [52, 53] found among multiple related species. This question is particularly important for the goal preservation in self-improving intelligent software, as we want to make sure that future generations of the system are motivated to work on the same problem [29]. As AI goes through the RSI process and becomes smarter and more rational it is likely to engage in a de-biasing process removing any constraints we programmed into it [8]. Ideally we would want to be able to prove that even after recursive self-improvement our algorithm maintains the same goals as the original. Proofs of safety or correctness for the algorithm only apply to particular source code and would need to be rewritten and re-proven if the code is modified, which happens in RSI software many times. But we suspect that re-proving slightly modified code may be easier compared to having to prove safety of a completely novel piece of code.

We are also interested in understanding if RSI process can take place in an isolated (leakproofed [54]) system or if interaction with external environment, internet, people, other AI agents is necessary. Perhaps access to external information can be used to mediate speed of RSI process. This also has significant implications on safety mechanisms we can employ while experimenting with early RSI systems [55-63]. Finally, it needs to be investigated if the whole RSI process can be paused at any point and for any specific duration of time in order to limit any negative impact from potential intelligence explosion. Ideally we would like to be able to program our Seed AI to RSI until it reaches certain level of intelligence, pause and wait for further instructions.

4 Conclusions

Recursively Self-Improving software is the ultimate form of artificial life and creation of life remains one of the great unsolved mysteries in science. More precisely, the

problem of creating RSI software is really the challenge of creating a program capable of writing other programs [64], and so is an AI-Complete problem as has been demonstrated by Yampolskiy [65, 66]. AI-complete problems are by definition most difficult problems faced by AI researchers and it is likely that RSI source code will be so complex that it would be difficult or impossible to fully analyze [51]. Also, the problem is likely to be NP-Complete as even simple metareasoning and metalearning [67] problems have been shown by Conitzer and Sandholm to belong to that class. In particular they proved that allocation of deliberation time across anytime algorithms running on different problem instances is NP-Complete and a complimentary problem of dynamically allocating information gathering resources by an agent across multiple actions is NP-Hard, even if evaluating each particular action is computationally simple. Finally, they showed that the problem of deliberately choosing a limited number of deliberation or information gathering actions to disambiguate the state of the world is PSPACE Hard in general [68].

This paper is a part of a two paper set presented at AGI2015 with the complementary paper being: "On the Limits of Recursively Self-Improving AGI" [69].

References

1. Turing, A.: Computing Machinery and Intelligence. Mind **59**(236), 433–460 (1950)
2. Good, I.J.: Speculations Concerning the First Ultraintelligent Machine. Advances in Computers **6**, 31–88 (1966)
3. Minsky, M.: Artificial Intelligence. Scientific American **215**(3), 257 (1966)
4. Burks, A.W., Von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press (1966)
5. Pearce, D.: The biointelligence explosion. In: Singularity Hypotheses, pp. 199–238. Springer (2012)
6. Omohundro, S.M.: The nature of self-improving artificial intelligence. In: Singularity Summit, San Francisco, CA (2007)
7. Waser, M.R.: Bootstrapping a structured self-improving & safe autopoietic self. In: Annual International Conference on Biologically Inspired Cognitive Architectures, Boston, Massachusetts, November 9, 2014
8. Hall, J.S.: Engineering utopia. Frontiers in Artificial Intelligence and Applications **171**, 460 (2008)
9. Mavrogiannopoulos, N., Kisserli, N., Preneel, B.: A taxonomy of self-modifying code for obfuscation. Computers & Security **30**(8), 679–691 (2011)
10. Anckaert, B., Madou, M., De Bosschere, K.: A model for self-modifying code. In: Camenisch, J.L., Collberg, C.S., Johnson, N.F., Sallee, P. (eds.) IH 2006. LNCS, vol. 4437, pp. 232–248. Springer, Heidelberg (2007)
11. Petrean, L.: Polymorphic and Metamorphic Code Applications in Portable Executable Files Protection. Acta Technica Napocensis, **51**(1) (2010)
12. Bonfante, G., Marion, J.-Y., Reynaud-Plantey, D.: A computability perspective on self-modifying programs. In: Seventh IEEE International Conference on Software Engineering and Formal Methods, pp. 231–239. IEEE (2009)
13. Cheng, B.H., et al.: Software engineering for self-adaptive systems: a research roadmap. In: Cheng, B.H., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) Software Engineering for Self-Adaptive Systems. LNCS, vol. 5525, pp. 1–26. Springer, Heidelberg (2009)

14. Ailon, N., et al.: Self-improving algorithms. SIAM Journal on Computing **40**(2), 350–375 (2011)
15. Yampolskiy, R., et al.: Printer model integrating genetic algorithm for improvement of halftone patterns. In: Western New York Image Processing Workshop (WNYIPW). IEEE Signal Processing Society, Rochester, NY (2004)
16. Yampolskiy, R.V., Ashby, L., Hassan, L.: Wisdom of Artificial Crowds—A Metaheuristic Algorithm for Optimization. Journal of Intelligent Learning Systems and Applications **4**(2), 98–107 (2012)
17. Yampolskiy, R.V., Ahmed, E.L.B.: Wisdom of artificial crowds algorithm for solving NP-hard problems. International Journal of Bio-Inspired Computation (IJBIC) **3**(6), 358–369
18. Ashby, L.H., Yampolskiy, R.V.: Genetic algorithm and wisdom of artificial crowds algorithm applied to light up. In: 16th International Conference on Computer Games: AI, Animation, Mobile, Interactive Multimedia, Educational & Serious Games, Louisville, KY, USA, pp. 27–32, July 27–30, 2011
19. Khalifa, A.B., Yampolskiy, R.V.: GA with Wisdom of Artificial Crowds for Solving Mastermind Satisfiability Problem. International Journal of Intelligent Games & Simulation **6**(2), 6 (2011)
20. Port, A.C., Yampolskiy, R.V.: Using a GA and Wisdom of Artificial Crowds to solve solitaire battleship puzzles. In: 17th International Conference on Computer Games (CGAMES), pp. 25–29. IEEE, Louisville (2012)
21. Omohundro, S.: Rational artificial intelligence for the greater good. In: Singularity Hypotheses, pp. 161–179. Springer (2012)
22. Anderson, M.L., Oates, T.: A review of recent research in metareasoning and metalearning. AI Magazine **28**(1), 12 (2007)
23. Yudkowsky, E.: Intelligence explosion microeconomics. In: MIRI Technical Report. www.intelligence.org/files/IEM.pdf
24. Heylighen, F.: Brain in a vat cannot break out. Journal of Consciousness Studies **19**(1–2), 1–2 (2012)
25. Turchin, V.F.: The concept of a supercompiler. ACM Transactions on Programming Languages and Systems (TOPLAS) **8**(3), 292–325 (1986)
26. Sotala, K.: Advantages of artificial intelligences, uploads, and digital minds. International Journal of Machine Consciousness **4**(01), 275–291 (2012)
27. Muehlhauser, L., Salamon, A.: Intelligence explosion: evidence and import. In: Singularity Hypotheses, pp. 15–42. Springer (2012)
28. Yudkowsky, E.: Levels of organization in general intelligence. In: Artificial General Intelligence, pp. 389–501. Springer (2007)
29. Chalmers, D.: The Singularity: A Philosophical Analysis. Journal of Consciousness Studies **17**, 7–65 (2010)
30. Nivel, E., et al.: Bounded Recursive Self-Improvement. arXiv preprint arXiv:1312.6764 (2013)
31. Nivel, E., Thórisson, K.R.: Self-programming: operationalizing autonomy. In: Proceedings of the 2nd Conf. on Artificial General Intelligence (2008)
32. Yudkowsky, E., Hanson, R.: The Hanson-Yudkowsky AI-foom debate. In: MIRI Technical Report (2008). http://intelligence.org/files/AIFoomDebate.pdf
33. Yampolskiy, R.V.: The Universe of Minds. arXiv preprint arXiv:1410.0369 (2014)
34. Hall, J.S.: Self-improving AI: An analysis. Minds and Machines **17**(3), 249–259 (2007)

35. Yampolskiy, R.V.: Efficiency Theory: a Unifying Theory for Information, Computation and Intelligence. Journal of Discrete Mathematical Sciences & Cryptography **16**(4–5), 259–277 (2013)
36. Gagliolo, M.: Universal search. Scholarpedia **2**(11), 2575 (2007)
37. Levin, L.: Universal Search Problems. Problems of Information Transmission **9**(3), 265–266 (1973)
38. Steunebrink, B., Schmidhuber, J.: A Family of Gödel Machine implementations. In: Fourth Conference on Artificial General Intelligence (AGI-11), Mountain View, California (2011)
39. Schmidhuber, J.: Gödel machines: fully self-referential optimal universal self-improvers. In: Artificial General Intelligence, pp. 199–226. Springer (2007)
40. Schmidhuber, J.: Gödel machines: towards a technical justification of consciousness. In: Adaptive Agents and Multi-Agent Systems II, pp. 1–23. Springer (2005)
41. Schmidhuber, J.: Gödel machines: self-referential universal problem solvers making provably optimal self-improvements. In: Artificial General Intelligence (2005)
42. Schmidhuber, J.: Ultimate cognition à la Gödel. Cognitive Computation **1**(2), 177–193 (2009)
43. Schmidhuber, J.: Completely self-referential optimal reinforcement learners. In: Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S. (eds.) ICANN 2005. LNCS, vol. 3697, pp. 223–233. Springer, Heidelberg (2005)
44. Schmidhuber, J.: Optimal ordered problem solver. Machine Learning **54**(3), 211–254 (2004)
45. Schmidhuber, J., Zhao, J., Wiering, M.: Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement. Machine Learning **28**(1), 105–130 (1997)
46. Schmidhuber, J.: A general method for incremental self-improvement and multiagent learning. Evolutionary Computation: Theory and Applications, 81–123 (1999)
47. Schmidhuber, J.: Metalearning with the Success-Story Algorithm (1997). http://people.idsia.ch/~juergen/ssa/sld001.htm
48. Schmidhuber, J.: A neural network that embeds its own meta-levels. In: IEEE International Conference on Neural Networks, pp. 407–412. IEEE (1993)
49. Younger, A.S., Hochreiter, S., Conwell, P.R.: Meta-learning with backpropagation. In: International Joint Conference on Neural Networks (IJCNN 2001). IEEE (2001)
50. Hochreiter, S., Younger, A., Conwell, P.: Learning to learn using gradient descent. In: Artificial Neural Networks—ICANN 2001, pp. 87–94 (2001)
51. Osterweil, L.J., Clarke, L.A.: Continuous self-evaluation for the self-improvement of software. In: Robertson, P., Shrobe, H.E., Laddaga, R. (eds.) IWSAS 2000. LNCS, vol. 1936, pp. 27–39. Springer, Heidelberg (2001)
52. Beck, M.B., Rouchka, E.C., Yampolskiy, R.V.: Finding data in DNA: computer forensic investigations of living organisms. In: Rogers, M., Seigfried-Spellar, K.C. (eds.) ICDF2C 2012. LNICST, vol. 114, pp. 204–219. Springer, Heidelberg (2013)
53. Beck, M., Yampolskiy, R.: DNA as a medium for hiding data. BMC Bioinformatics **13**(Suppl. 12), A23 (2012)
54. Yampolskiy, R.V.: Leakproofing Singularity - Artificial Intelligence Confinement Problem. Journal of Consciousness Studies (JCS) **19**(1–2), 194–214 (2012)
55. Majot, A.M., Yampolskiy, R.V.: AI safety engineering through introduction of self-reference into felicific calculus via artificial pain and pleasure. In: 2014 IEEE International Symposium on Ethics in Science, Technology and Engineering. IEEE (2014)
56. Yampolskiy, R., Fox, J.: Safety Engineering for Artificial General Intelligence, pp. 1–10. Topoi (2012)

57. Yampolskiy, R.V., Fox, J.: Artificial general intelligence and the human mental model. In: Singularity Hypotheses: A Scientific and Philosophical Assessment, p. 129 (2013)
58. Sotala, K., Yampolskiy, R.V.: Responses to catastrophic AGI risk: A survey. Physica Scripta. **90**, December 2015
59. Yampolskiy, R.V.: What to do with the singularity paradox? In: Müller, V.C. (ed.) Philosophy and Theory of Artificial Intelligence. SAPERE, vol. 5, pp. 397–413. Springer, Heidelberg (2012)
60. Yampolskiy, R., Gavrilova, M.: Artimetrics: Biometrics for Artificial Entities. IEEE Robotics and Automation Magazine (RAM) **19**(4), 48–58 (2012)
61. Yampolskiy, R., et al.: Experiments in Artimetrics: Avatar Face Recognition. Transactions on Computational Science XVI, 77–94 (2012)
62. Ali, N., Schaeffer, D., Yampolskiy, R.V.: Linguistic profiling and behavioral drift in chat bots. In: Midwest Artificial Intelligence and Cognitive Science Conference, p. 27 (2012)
63. Gavrilova, M., Yampolskiy, R.: State-of-the-Art in Robot Authentication [From the Guest Editors]. Robotics & Automation Magazine, IEEE **17**(4), 23–24 (2010)
64. Hall, J.S.: VARIAC: an Autogenous Cognitive Architecture. Frontiers in Artificial Intelligence and Applications **171**, 176 (2008)
65. Yampolskiy, R.V.: Turing test as a defining feature of ai-completeness. In: Yang, X.-S. (ed.) Artificial Intelligence, Evolutionary Computing and Metaheuristics. SCI, vol. 427, pp. 3–17. Springer, Heidelberg (2013)
66. Yampolskiy, R.V.: AI-Complete, AI-Hard, or AI-Easy–Classification of problems in AI. In: The 23rd Midwest Artificial Intelligence and Cognitive Science Conference, Cincinnati, OH, USA (2012)
67. Schaul, T., Schmidhuber, J.: Metalearning. Scholarpedia **5**(6), 4650 (2010)
68. Conitzer, V., Sandholm, T.: Definition and complexity of some basic metareasoning problems. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI), Acapulco, Mexico, pp. 1099–1106 (2003)
69. Yampolskiy, R.V.: On the limits of recursively self-improving AGI. In: The Eighth Conference on Artificial General Intelligence, Berlin, Germany, July 22–25, 2015