Putnam's Rocks Are Clocks

Ramón Casares

ORCID: 0000-0003-4973-3128

Putnam proved that "every ordinary open system is a realization of every abstract finite automaton", showing that computing is meaningless. Analyzing a simpler version of his proof, we conclude that giving a meaning to a computation requires computing, which is meaningless, starting a recursion.

Keywords: *Putnam, implementation, computing, meaning.*

§1 Putnam's theorem

¶1 · Putnam (1988), pages 121–125, proved the following theorem:

> *Every ordinary open system is a realization of every abstract finite automaton.*

¶2 · An abstract finite automaton, also known as finite-state automaton, is a mathematical concept, so it does not need further specifications, see for example Mealy (1955), but an ordinary open system requires some additional physical stipulations in order to fit it into a mathematical theorem. So before the proof, Putnam explains the two Physical Principles that he needs: the Principle of Continuity and the Principle of Noncyclical Behavior. Putnam assures us that the Principle of Noncyclical Behavior is true for any physical system affected by a clock, and that every open system is affected.

¶3 · We will proceed otherwise. Instead of assuming that every ordinary open system is affected by a clock, we will use a clock as system. This way we can completely ignore the physics of the theorem, including the Principle of Noncyclical Behavior, and we can instead focus on its mathematical content. So we will restate Putnam's theorem in a simpler way by using a simple clock instead of the ordinary open system, we will prove it following his trail, and we will analyze the consequences.

¶4 · Putnam's theorem shows that computing by itself is meaningless, and our analysis shows that giving a meaning to a computation requires computing, which is meaningless as shown by Putnam's theorem. Therefore, Putnam's theorem is just a step in a recursion. Finally, I will propose a way out of the recursion.

This is DOI: 10.6084/m9.figshare.5450278.v2, version 20171016.

Any comments on it to papa@ramoncasares.com are welcome.

§2 Simpler theorem

§2.1 Theorem

¶1 · We will prove the simpler theorem:

Every simple clock is a realization of every abstract finite automaton.

§2.2 Definition

¶1 · A simple clock is one that implements the successor function, so its starting state is 0, and the next state is the next natural number. So for any simple clock $s_0 = 0$, $s_1 = 1$, $s_2 = 2$, and in general $s_n = n$.

§2.3 Proof

¶1 · Any finite string of states, generated by any abstract finite automaton, can be realized by any simple clock using the following rule: assign the disjunction of the positions where it appears to each state in the string. For example, for string $ABABABA$, define $A = 0 \vee 2 \vee 4 \vee 6$ and $B = 1 \vee 3 \vee 5$. Using these definitions, the clock realizes the string. Q.E.D.?

§2.4 Analysis

¶1 · As you can see, our proof has distilled the essence of Putnam's proof, avoiding any possible distraction. The analysis is now easier.

¶2 · The simple clock is perhaps the simplest clock, but the only requirement that any clock has to fulfill is that it cannot repeat any state. If, for example $s_j = s_n$, where $j \neq n$, that is, if state at time j were repeated at time n, not necessarily consecutive, then it could not realize every finite-state automaton, but only those that were at the same state at times j and n. The simple clock does not repeat any state, thus providing a unique index. In Putnam's proof, the Principle of Noncyclical Behavior grants a unique index.

¶3 · Using the simple clock, the rule used in our proof defines a partial function f that assigns a state of the finite-state automaton to the first natural numbers. Following the example, $f(0) = A$, $f(1) = B$, $f(2) = A$, $f(3) = B$, $f(4) = A$, $f(5) = B$, and $f(6) = A$. We will call this function the interpreting function.

¶4 · So the simple clock generates a series $0, 1, 2, 3, 4, 5, 6$, which is transformed by the interpreting function into the series $f(0), f(1), f(2), f(3), f(4), f(5), f(6)$ that realizes the string generated by the finite-state automaton $ABABABA$.

¶5 · By the simple clock definition in §2.2, we know that the next state of the simple clock will be $s_7 = 7$, but we cannot use the simple clock to foresee what will be the next state of the finite-state automaton. To make that prediction we will need to know the value of the interpreting function for number 7, that is, $f(7)$. And in general, to know the state of the finite-state automaton at any instant, we will need to know the total version of the interpreting function, that is, the value of $f(n)$ for any $n \in \mathbb{N}$.

¶6 · What indeed realizes the finite-state automaton is not the simple clock, which only provides a temporal index, but the interpreting function, which is not just a device of the proof, as it seemed to be. In other words, what the proof shows is this slightly different three-part theorem: *Using suitable interpreting functions, every simple clock is a realization of every abstract finite automaton.* Putnam's theorem requires the same amendment.

¶7 · Since Turing's (1937) proof, we know that realizing a function is equivalent to computing the function, where 'realizing' means 'effectively calculating' as in Church's (1935) thesis. As this is true for any function, it applies also to every interpreting function:

Realizing any interpreting function is computing it.

§3 Discussion

¶1 · See Chalmers (1996), §4, page 317, for another simplification of Putnam's theorem. You can find there a proof that: "Every physical system containing a clock and a dial will implement every inputless FSA [finite-state automaton]." But, instead of amending the theorem as we do, that is, by ascribing the mind to the proof reader who realizes the interpreting function, he concludes that, as a clock and a dial cannot have a mind, his proof shows that no implementation of an inputless finite-state automaton has a mind.

¶2 · Similarly, in §5, Chalmers concludes that no implementation of a finite-state automaton with input and output has a mind because of the simplicity of implementing any of them. In §6, page 324, he explains us that:

> *The trouble is that the internal states of these FSAs are monadic, lacking any internal structure, whereas the internal states of most computational and cognitive systems have all sorts of complex structure.*

Then his solution is the combinatorial state automaton, or CSA, which is a finite-state automaton but with complex states. Therefore, according to Chalmers, a rock has not a mind because it implements an inputless finite-state automaton, and a human brain has a mind because it implements a combinatorial state automaton.

¶3 · But I am not convinced by Chalmers solution. As he writes in page 325:

> *For every CSA, there is an FSA that can simulate it.*

Now suppose that evolution produces a new subspecies, call it *Homo sapiens FSA*, that is exactly as our actual species, call the corresponding subspecies *Homo sapiens CSA*, except that the *Homo sapiens FSA* brain implements an FSA that simulates the CSA that the *Homo sapiens CSA* brain implements. Then, following Chalmers, although they would be nearly indistinguishable, the *Homo sapiens FSA* would be mindless, while the *Homo sapiens CSA* is mindful.

¶4 · To me, most arguments relying on complexity are suspicious, from the derogatory 'this is too complex for you to understand' to Chalmers' 'mind is an effect of implementation complexity'. But, what is even more relevant here is that Chalmers does not show that complexity is independent of the observer, so we can think instead that implementation complexity depends on the computing resources of the observer.

¶5 · Examining the wall argument by Searle (1992), pages 208–209, which is similar to Putnam's theorem, Blackmon (2013) argues that program implementation is a three-place relation: physical things, programs, and mappings, where the mappings are our interpreting functions. This way he avoids the problems that Putnam, Searle and Chalmers' two-place implementation face. Giunti (2017) also defines implementation by seeing a computational system as a three-part object: a physical part, a mathematical part, and an interpretation.

¶6 · So we have a naïve epistemology that assumes that reality is translated identically into knowledge, and according to which truth is a two-place relation: fact and expression.

But, when the assumption of naïve epistemology breaks, we need a three-place relation that takes into consideration the agent who expresses the fact: fact, expression, and agent. We can call this three-place epistemology subjectivism because it can be derived from the epistemology of Kant: reality, knowledge, and subject.

¶7 · Our analysis confirms that implementation is a three-place relation: physical device, mathematical model, and interpreting function. But our analysis also reveals an issue: that the interpreting function of implementation has to be implemented, too. It seems that we go in circles.

reality	knowledge	subject
fact	expression	agent
thing	program	mapping
physical device	mathematical model	interpreting function

§4 Conclusion

¶1 · An idea behind Putnam's theorem is that any computation is meaningless, except when it is interpreted by a mind. When you press, in this order, the four buttons labeled '2', '+', '3', and '=', of your calculator, its screen lights up some segments that you interpret as the number 5. You can say that your calculator is adding numbers, but it is only because you are giving some meanings to the labeled buttons and to the lighted up screen segments. In this example, you are the mindful agent who interprets that the physical ordinary open system that is your calculator is realizing the mathematical abstract finite automaton that defines addition. The same happens to a full programmable computer, as shown by Searle (1980 and 1992) with his Chinese room and wall arguments.

¶2 · In the case of our proof, the interpreting function plays the the rôle of the interpreting mind. What we have found in our analysis is that the interpreting mind has to perform computations, too. Then Putnam's theorem and Searle's Chinese room and wall arguments show that computing by itself is meaningless, and our analysis shows that giving a meaning to a computation requires computing, which is meaningless, starting a recursion.

¶3 · To me, this recursion halts whenever a computation resolves a problem, because then the problem provides the meaning, so I would say that any computation is meaningful for its resolver, regardless whether it is mindful or mindless. Therefore, for those who can agree with me in this, all brain computations resolving the survival problems of living beings are meaningful.

Acknowledgments

¶1 · I am very grateful to Mark Bishop, James Blackmon, David Chalmers, and Marco Giunti for valuable discussions and commentary on the first version of this paper.

References

Blackmon (2013): James Blackmon, "Searle's Wall"; in *Erkenntnis*, vol. 78, no. 1, pp 109–117, February 2013, DOI: 10.1007/s10670-012-9405-4.

Chalmers (1996): David J. Chalmers, "Does a Rock Implement Every Finite-State Automaton?"; in *Synthese*, vol. 108, no. 3, pp. 309–333, 1996, DOI: 10.1007/BF00413692.

Church (1935): Alonzo Church, "An Unsolvable Problem of Elementary Number Theory"; in *American Journal of Mathematics*, vol. 58, no. 2, pp. 345–363, April 1936, DOI: 10.2307/2371045. Presented to the American Mathematical Society, April 19, 1935.

Giunti (2017): Marco Giunti, "What is a Physical Realization of a Computational System?"; in *Isonomia Epistemologica*, vol. 9, pp. 177–192, 2017, http://isonomia.uniurb.it/reasoning-metaphor-and-science/.

Mealy (1955): George H. Mealy, "A Method for Synthesizing Sequential Circuits"; in *Bell System Technical Journal*, vol. 34, no. 5, pp. 1045–1079, September 1955, DOI: 10.1002/j.1538-7305.1955.tb03788.x.

Putnam (1988): Hilary Putnam, *Representation and Reality*; The MIT Press, Cambridge, MA, 1988, ISBN: 978-0-262-66074-7.

Searle (1980): John R. Searle, "Minds, Brains, and Programs"; in *The Behavioral and Brain Sciences*, vol. 3, no. 3, pp. 417–424, September 1980, DOI: 10.1017/S0140525X00005756.

Searle (1992): John R. Searle, *The Rediscovery of the Mind*; The MIT Press, Cambridge, MA, 1992, ISBN: 978-0-262-69154-3.

Turing (1937): A. M. Turing, "Computability and λ-Definability"; in *The Journal of Symbolic Logic*, vol. 2, no. 4, pp. 153–163, December 1937, DOI: 10.2307/2268280.