A Minimal Intervention Definition of Reverse Engineering a Neural Circuit

Keerthana Gurushankar and Pulkit Grover
Department of Electrical & Computer Engineering, and Neuroscience Institute, Carnegie Mellon University

Abstract—In neuroscience, informal notions of what it means to reverse engineer a system are commonly used, e.g., being able to simulate a system and/or predict its response. A recent work of Jonas & Kording examines a microprocessor using commonly used neuroscience techniques, and suggests that they are inadequate. Part of the difficulty, as a previous work of Lazebnik noted, lies in lack of formal language. Motivated by these papers, in this extended essay, we provide a theoretical framework for defining reverse engineering of computational systems for neuroscience contexts. Inspired by recent clinical efforts of localized interventions to alter the function of the neural circuitry to treat disorders, and by Lazebnik’s viewpoint that understanding a system means you can “fix it”, we propose the following requirement on reverse engineering: once an agent claims to have reverse-engineered a neural circuit, they subsequently need to be able to: (a) provide a minimal set of interventions to change the input/output (I/O) behavior of the circuit to a desired behavior; (b) arrive at this minimal set of interventions while operating under some constraints, e.g., limited memory, or access to only observations of the system, etc. to rule out brute-force approaches. We show that, under certain assumptions, this reverse engineering goal falls within the class of undecidable problems, by connecting our problem with Rice’s theorem in theoretical computer science. Next, we examine some canonical computational systems and reverse engineering goals (as specified by desired I/O behaviors) where reverse engineering can indeed be performed. In complementary works, we examine how information-theoretic approaches can provide insights into the problem.

I. OVERVIEW

Our recent works [1], [2] use information theoretic approaches to address a reverse engineering problem relevant to neuroscience: finding minimal interventions on a given circuit to change its input/output behavior to a desirable one. This paper provides the foundation of these studies. We provide a formal problem statement, discuss its motivation, and examine the difficulty of the question from a theoretical computer science perspective.

Our formulation is motivated broadly from two works in this century that point to the need for formal definitions and rigor in understanding neural computation. The essay of Lazebnik [3], provocatively titled “Can a biologist fix a radio,” emphasizes on the need for a formal language to describe elements and questions within biology so that there is reduced ambiguity or vagueness, and clear (falsifiable) predictions are made. This need is becoming increasingly evident in attempts to reverse engineer the brain. While neural recording and stimulation technology is advancing rapidly1, and techniques for analyzing data with statistical guarantees have also expanded rapidly, the techniques do not provide satisfying answers for understanding the system [5], [6]. This is most evident in the strikingly detailed work of Jonas and Kording [5]2, which use

1 A “Moore’s law of neural recordings” is being witnessed: the number of neurons being recorded simultaneously is increasing exponentially [4].
2 Titled “Could a neuroscientist understand a microprocessor?”, [5] follows in the footsteps of Lazebnik’s, but also tests popular techniques from computational neuroscience. See also the Mus Silicium project [7].

an early but sophisticated microprocessor, MOS 6502, instead of Lazebnik’s radio. They examine this microprocessor under 3 different “behaviors” (corresponding to 3 different computer games, namely, Donkey Kong, Space Invaders, and Pitfall), and conclude that “… current analytic approaches in neuroscience may fall short of producing meaningful understanding of neural systems, regardless of the amount of data”. The work also underscored the need for rigorous testing of tools on simulated data prior to application on real data for obtaining inferences. Because they focus on concrete implementations and a fully specified and simple system, they conclude that they should obtain an understanding that “guides us towards the descriptions” commonly used in computer architecture (e.g., an Arithmetic Logical Unit consisting of simple units such as adders, a memory). Subjective definitions of reverse engineering have been explored elsewhere as well (e.g. [8], [9]).

Inspired by [3], [5], we ask the normative question: what end-goal for reverse engineering should the neuroscientists aim for? Our main intellectual contribution in this context can be summarized in two pieces: a) Viewing reverse engineering as faithful summarization, i.e., one needs to represent the computation not just faithfully but also economically; and b) Specifying what may constitute faithful representation of a computation in the context of neuroscience. Specifically, we take an minimal-interventional view of faithful representation, as explained below.

Reverse engineering is faithful summarization: The act of modeling/abstracting itself is compression, as good models tend to preserve the essence of the phenomenon/aspect of interest, discarding the rest [10]. This is also reflected in neuroscience-related works [11]. Literature in Algorithmic Information Theory, which uses Kolmogrov complexity (minimal length of a code to compute a function) to quantify the degree of compression, has also been connected to understanding the computation [12]. E.g., a reverse engineering agent (human or artificial) should be able to compress the description of the computational system in a few bits. The degree to which the description can be compressed, while still maintaining a faithful representation, quantifies the level or degree of understanding (i.e., reverse engineering). This compression rules out, for instance, brute-force approaches that store a simulation of the entire computational system as reverse engineering (discussed further in Section II).

What constitutes faithful representation: How do we quantify faithfulness of a representation? We believe it is important to not just preserve the input/output (I/O) relationship, but also preserve how the function is computed, summarizing relevant information from the structure and architecture of the network and the function computed at each of the nodes (e.g., the structure of the Fast Fourier transform, FFT Butterfly network, considered in Section V, is integral to how the FFT
is often implemented). In other words, preserving only the I/O relationship misses the point of how the computation is carried out (it preserves, exclusively, what function is implemented, but not how). Motivated by operational goals of understanding implementation as a way of understanding how the computation is performed, we impose an interventional requirement on faithful representations, namely, that a representation is faithful if it enables predicting minimal interventions that change the I/O behavior of the system from the existing behavior to another desired behavior. Our emphasis on minimal interventions is because we want to rule out approaches that change the entire system to another system (i.e., those that only rely on the I/O relationship and not the structure/implementation, e.g., an approach that replaces the entire system with one that has a desirable I/O behavior might not be a minimal intervention).

Tying the two aspects above together, we arrive at our definition of reverse engineering (more intuition in Section II, formally stated in Section III). Informally, one must be able to summarize the description using just a few bits, and this description should suffice for minimal interventions to change the I/O relationship to a desired one.

Our interventional definition is not without precedence. Indeed, a classical (if informal) view of understanding a system requires that one must be able to break it into pieces and put it back together, or, in Lazebnik’s words [3], “fix” it. Some existing approaches in explainable/interpretable machine-learning also use interventions to understand the system, e.g., influence of features on the output [13]. This might offer an achievability of reverse engineering, but our work is distinct in that it attempts to define explainability in an interventional sense. Here, our goal is one of editing the network (and not just the features) to demonstrate understanding. Interventionist accounts of explanations have been discussed in philosophy of science. Woodward [14] argues in support of explanations that describe not only the I/O behavior of the system, but also the behavior after interventions. In the context of neuroscience, Craver [11], among others, separates “explanatory models” from “phenomenally adequate”. Whereas phenomenally adequate models might only describe or summarize the phenomenon of interest, explanatory models should also allow a degree of control and manipulation.

These views are well aligned with ours. Additionally, our work (specifically, the minimal interventions aspect) is motivated by advances in neural engineering and clinical efforts in treating disorders. Recent efforts have succeeded in engineering minimally-invasive implantable systems (e.g. neural dust, nanoparticles, injectable electronics [15, 16]) (even noninvasive techniques are increasing in their precision [17]). Recent clinical efforts in humans have involved chronic (i.e., long-term) implantation of electrodes for treating depression [18], obsessive-compulsive disorder (OCD) [19], addiction [20], obesity [21], etc., which are all disorders of the reward network discussed Appendix C. One clinical end-goal is to manipulate this biological circuit with minimal interventions.

In explainable AI literature, there is an acknowledgment that being able to propose interventions is a way of demonstrating understanding of a decision-making system [9, 22, 23], although much of this body of work is focused on interventions on the feature space [24, 25] or individual data points [26, 27], rather than inside the computational network. In AI, it is often not required for explanations to be at a physical implementation level. In neuroscience, as noted here, explanations tied to the implementation can help with interventions for treating disorders (specially with recent advances in neuroengineering).

**What this work accomplishes.** The main contribution of this paper is 3-fold, (i) the reverse-engineering definition itself, stated formally in Section III. (ii) An undecidability result: In the spirit of formal treatments, even under optimistic assumptions on what can be learned about the system through observations and interventions, we obtain a hardness/impossibility result, showing that a sub-class of the general set of reverse engineering problems is undecidable, i.e., no agent which is itself a Turing machine can provide a desirable reverse engineering solution for arbitrarily chosen problems for our minimal-interventions definition. This result is obtained by connecting Rice’s theorem from theoretical computer science [28] with our reverse engineering problem, and is the first connection drawn between neuroscience and Rice’s theorem. Further, to illustrate how this result about the undecidability of reverse engineering is not merely an artifact of our chosen definitions, we also include alternative plausible definitions of reverse engineering, and proofs of their undecidability in Appendix A; (iii) Examples: In Section V, we illustrate that this goal is attainable in interesting (if toy) cases, by using examples of simple computational systems, and describing their reverse engineering solutions. In Appendix C, we include discussion on an an exemplar neural circuit: the reward network. There, we overview the state of understanding of this exemplar circuit and discuss what it may lack from our reverse engineering perspective. We conclude in Appendix D, including limitations of our work.

**Place within information theory scope and literature:** In Section II, we provide a more detailed literature review to help position this work in the neuroscience context. Within the IT context, our main contribution is the definitions and a connection with models used in neuroscience (see, e.g. models in [29], [30], etc.), specifically, the aspect of compression in our definition, as well as how information-theoretic approaches are brought to bear on this problem. Concretely, in complementary works [1], [2], we adopt this framework. There, we impose the additional requirement that the reverse engineering be performed using just observations of the system (measurements at each node and edge), without any interventions, but from the I/O relationship, we only require the system to have high accuracy and/or less bias. This allows us to formally examine neuroscience questions using IT techniques developed in [31], [32], [29]. The undecidability results here also fall out of making this formal connection. More broadly, modifications on our approach and models can pave the way to more formal treatment of neuroscience problems from an IT lens, including algorithmic advances on problems of reverse engineering.

Note that our formulation differs from the information-bottleneck approach [33] and adversarial weight perturbations [34], neither of which are aimed at performing minimal interventions. The focus of the current work is the statement of the problem and an examination of its difficulty.

**II. BACKGROUND AND RELATED NEUROSCIENCE WORK**

Explicitly or not, the question posed here connects with all works in neuroscience. Thus, rather than task ourselves with
the infeasible goal of a thorough neuroscience survey, we strive to illustrate the evolution of the relevant neuroscience discussion.

Perhaps the simplest reverse-engineering of a computational system is being able to “simulate” the I/O behavior of the system (see Introduction of [5]). E.g., cochlear and retinal prostheses attempt to replace a (nonfunctional) neural system with a desirable system with “healthy” I/O behavior (see also [35], [36] for examples of such attempts for sensory processing and memory, respectively). This “black-box” way of thinking may suffice for understanding what is being computed, but not how. To describe how a computation is being performed, one might seek to describe the input-output behavior of individual elements of computation (which could be as fine-grained as compartments of a single neuron’s membrane, or a neuron itself, or a collection of neurons). There is a compelling argument that even this component-level simulation is insufficient. E.g., Gao and Ganguli [6], in their work on required minimal measurements in neuroscience, note that while we can completely simulate artificial neural networks (ANNs), most machine-learning researchers would readily accept that we do not understand them. This led Gao and Ganguli to ask: “...can we say something about the behavior of deep or recurrent ANNs without actually simulating them in detail?” (see related field of “explainable machine-learning” [8], [37]). That is, a component-level understanding can miss an understanding at an intuitive level.

To state what a more comprehensive understanding of a computational system could look like, inspired by the visual system, cognitive scientist David Marr proposed “3 levels of analysis” [38]: computational, algorithmic, and implementation. At the lowermost, implementation level, is the question of how a computation is implemented in its hardware. Above that, at the algorithmic level, the question, stated informally by Marr, is what algorithm is being implemented, e.g., how it represents information and modulates these representations. Finally, at the highest level is the problem being solved itself. We refer the reader to [39] for some of the recent discussions on Marr’s levels. Gao and Ganguli write in agreement, with subtle differences: “understanding will be found when we have the ability to develop simple coarse-grained models, or better yet a hierarchy of models, at varying levels of biophysical detail, all capable of predicting salient aspects of behavior at varying levels of resolution”. While influential and useful, Marr’s and Gao/Ganguli’s descriptions are too vague to quantify reverse engineering in a formal sense.

An exciting alternative approach was recently proposed by Lansdell and Kording [40]. Motivated by lack of satisfactory understanding of ANNs, their approach is to change the goals. They ask the question: can we learn the rules of learning, and that, at the algorithmic level, the question, stated informally by Marr, is what algorithm is being implemented, e.g., how it represents information and modulates these representations. Finally, at the highest level is the problem being solved itself. We refer the reader to [39] for some of the recent discussions on Marr’s levels. Gao and Ganguli write in agreement, with subtle differences: “understanding will be found when we have the ability to develop simple coarse-grained models, or better yet a hierarchy of models, at varying levels of biophysical detail, all capable of predicting salient aspects of behavior at varying levels of resolution”. While influential and useful, Marr’s and Gao/Ganguli’s descriptions are too vague to quantify reverse engineering in a formal sense.

An exciting alternative approach was recently proposed by Lansdell and Kording [40]. Motivated by lack of satisfactory understanding of ANNs, their approach is to change the goals. They ask the question: can we learn the rules of learning, and that, at the algorithmic level, the question, stated informally by Marr, is what algorithm is being implemented, e.g., how it represents information and modulates these representations. Finally, at the highest level is the problem being solved itself. We refer the reader to [39] for some of the recent discussions on Marr’s levels. Gao and Ganguli write in agreement, with subtle differences: “understanding will be found when we have the ability to develop simple coarse-grained models, or better yet a hierarchy of models, at varying levels of biophysical detail, all capable of predicting salient aspects of behavior at varying levels of resolution”. While influential and useful, Marr’s and Gao/Ganguli’s descriptions are too vague to quantify reverse engineering in a formal sense.

As discussed in Section I, complementary to these lines of thought, we take a fundamentally interventional view of reverse engineering. We also strive, in the established information-theoretic tradition, to state the problem formally, and then observe fundamental limits and achievabilities. This goal is challenging, to say the least, but efforts in this direction are needed to ground the questions in neuroscience concretely.

III. OUR MINIMAL INTERVENTION DEFINITION OF REVERSE ENGINEERING

Overview of our definition and rationale for our choices: We allow the agent performing the reverse engineering to specify several classes of desirable I/O relationships. To constrain the agent from using brute-force approaches, if the agent claims to have successfully reverse engineered the system, it must be able to produce a Turing machine that requires only a limited number of bits to describe. This Turing machine should be able to take a class of desirable I/O relationships as input, and provide as output a set of interventions that change the I/O relationship to one of the desirable ones within this class. The rationale for the requirement on the agent to provide a Turing machine is that it is a complete description of the summarization. An informal “compression” to a certain number of bits could hide the cost of encoding and decoding, or of some of the instructions in executing the algorithm. The rationale of allowing any of a class of I/O behaviors as an acceptable solution is that it allows for approximate solutions or choosing one among solutions that are (nonuniquely) optimal according to some criteria (e.g., in the reward circuitry which drives addiction, discussed in Section V, any I/O behavior that eliminates the reward of an addictive stimulus might suffice).

In addition, we allow the Turing machine to have a few accesses to the computational system where it can perform interventions and observe the changed I/O relationship. While this still disallows brute-force approaches, it enables lowering the bar on what is required for reverse engineering.

These definitions are there to lay down a formal framework in which we can obtain results. They can easily be modified. In arriving at this reverse engineering solution (i.e., in generating the Turing Machine), we allow the agent to access the “source code” of the computational system C. This might appear to be an optimistic assumption (indeed it is so) as it might require noiseless measurements everywhere, and possibly causal interventions, which current neuroengineering techniques are very far from. The definition can readily be modified to include access to limited noisy observations, which will only make the reverse engineering harder. Note that with “Moore’s law of neural recording,” it is conceivable that each node and edge can indeed be recorded in the distant (or nearby) future [4]. As another example, while we assume, for simplicity, that communication happens at discrete time-steps, this assumption can be relaxed for some of our results, e.g., our undecidability result in Section IV because it only makes the reverse engineering problem harder. Similarly, equipping the system with an additional external memory (e.g., the setup in [41]) also makes the reverse engineering problem harder.

A. System model

Definition 1 (Computational System and Computation). A computational system C is defined on a finite directed graph $\mathcal{G}(V,E)$. $V$ is a collection of nodes connected using directed edges $E$. The computation uses symbols in a set $S \subseteq \mathbb{R}$ (called the “alphabet” of C), where $0 \in S$. Each node $v$ stores a value in $S$ (initialized to any fixed $s \in S$). The computational input

3We acknowledge that I/O behavior can also have more or less understandable descriptions, e.g. machine-learning models of different complexity approximating the same I/O relationship. Thus a black-box way of describing I/O relationships has more nuance to it than is discussed here.
is a finite-length string of symbols in $S$. The computation starts at time $t = 0$ and happens over discrete time steps. At each time step, the $i$-th node, computes a function on a total of $n_i$ symbols, which includes (i) symbols stored in each node from which it has incoming edges (called “transmissions received from” the nodes they are stored in), (ii) the symbol stored in the node itself, and (iii) at most one symbol from the computational input. The node output at any time step, also a symbol in $S$, replaces the stored value. That is, the $i$-th node computes a function $S^{n_i} \rightarrow S$, mapping the $n_i$ symbols from the previous time instant (including nodes with incoming edges, the locally stored value, and the computation input) to update its stored value. The stored values across all nodes collectively form the “state” of the system at each time instant. A set of nodes are designated as the output nodes, and their first nonzero transmissions are together called the output of the computation.

We call this description of $C$, with $\mathcal{G}$ and the functions computed at the nodes the “source code” of $C$. This definition is inspired by similar ones in information theory and theory of computation [42], [43], and recent use in neuroscience [29].

**Definition 2** (Input/Output (I/O) relationship of $C$). The input-output relationship (I/O relationship) of $C$ is the mapping from the inputs to $C$ to the outputs of $C$.

**Definition 3** (Interventions on $C$). A single intervention on $C$ modifies the function being computed at exactly one of the nodes in $C$ at exactly one time instant.

An intervention will commonly change $C$’s I/O relationship.

**B. Definition of reverse engineering**

As discussed, our definition in essence is about making the system do what you want it to do. One way to view this, consistent with “fixing” the system, is by modifying the system $C$, we should be able to get the I/O relationship we desire.

Some notation: we use $H = \{ F_p \}_{p \in P}$ (for a countable index set $P$) to denote a collection of sets $F_p$ where each $F_p$ is a set of I/O relationships obtainable by multiple interventions on $C$. Intuitively, each element $F \in H$ represents a set of I/O relationships that are “equivalent” from the perspective of the end-goal of interventions on $C$. For instance, they could all approximate a desirable I/O relationship. As an illustration for the reward network, say $H = \{ F_1, F_2 \}$, where $F_1$ is the set of I/O relationships of unhealthy addiction, whereas $F_2$ might represent I/O relationships of healthy motivation.

To perform these interventions, we now define an agent $A$, whose goal is to generate a Turing machine that takes as input an index $p$, and provides as output the necessary interventions on $C$ to attain a desirable I/O relationship $g \in F_p$.

**Definition 4** (Reverse Engineering Agent $A$ and $M$-bit summarization). An agent $A$ takes as input the source-code of $C$ and $H$, a collection of sets of I/O relationships, and outputs a Turing-machine $TM_{C,M,H,Q}$ which is described using no more than $M$-bits. We refer to $TM_{C,M,H,Q}$ as an $M$-bit summarization of $C$. $TM_{C,M,H,Q}$ takes as input $p \in P$. Additionally, $TM_{C,M,H,Q}$ also has access to an oracle to which it can input up to $Q$ different sets of multiple interventions on $C$, and $p' \in P$. For each set of interventions, the oracle returns back whether the resulting I/O relationship for a set of multiple interventions lies in $F_{p'} \in H$. For any input $p \in P$, $TM_{C,M,H,Q}$ outputs a set of interventions $Z$ on $C$. It can also declare “no solution”.

Inspired by bounded-rationality approaches in economics and game theory [44], [45], [46], the $M$-bit summarization can enforce a constraint on $A$ that disallows brute-force approaches, e.g., where $A$ simply stores the changes in I/O relationships for all possible sets of interventions, and for a given reverse-engineering goal, simply retrieves the solution from the information it has stored. We now arrive at our definition of reverse engineering.

**Definition 5** ($(H, L, M, Q)$-Reverse Engineering). Consider a computational system $C$ with an I/O relationship described by $f(\cdot)$. Let $A$ be an agent that is claimed to have $(H, L, M, Q)$ reverse engineered $C$. Then, for a given $p \in P$ that is input to the Turing machine $TM_{C,M,H,Q}$ (which was generated by $A$), the output should be a set of interventions $Z$ of the smallest cardinality (if $|Z| \leq L$) that change the I/O relationship from $f(\cdot)$ to any $g \in F_p$ (but not necessarily for all $g \in F_p$). If no such $g \in F_p$ exists, then $TM_{C,M,H,Q}$ should declare “no solution”, i.e., no such set of ($L$ or fewer) interventions exists.

**IV. UNDECIDABLE REVERSE ENGINEERING PROBLEMS**

Reverse engineering is not undecidable for every class of $C$’s. It requires the class to be rich enough. Below, we first prove a result on how rich the class needs to be for it to be Turing-equivalent. Following this result, we use Rice’s theorem [28], [47] to make a formal connection with reverse engineering, proving in Theorem 3 that for set of $C$’s that use an $S$ of infinite cardinality, and computable functions at each node, the reverse engineering in Definition 5 is undecidable for nontrivial $H$’s, i.e., no agent $A$, that is itself a Turing Machine, can provide a reverse engineering solution for every $C$ in this class for any $L \geq 0$, any $M$ (including $M = \infty$), and $Q = 0$. Our undecidability result (Theorem 3, which uses Theorem 1.2 that is proven for a more limited set of $C$’s) is for a more restricted class (specifically, the $C$’s that can simulate “$\omega$-processor nets” of [48]) of computational systems than allowed in Definition 1. Hence, reverse engineering the broader class (for which Theorem 3 is stated) would only be harder (and hence is also undecidable).

**Theorem 1.** (1) If $|S|$ is finite, then the class of $C$’s in Def. 1 is equivalent to deterministic finite-state automaton (DFA).

(2) If $|S|$ is countably infinite (e.g. $\mathbb{Q}$) and all nodes compute computable functions, then the class of $C$’s is Turing equivalent.

(3) If the function at any node is uncomputable, then the class of $C$’s is super-Turing.

**Definition 6** (Nontrivial set of languages). The set of inputs accepted by a Turing machine is called its language. An input string is accepted by a Turing machine if the computation terminates in its accept state (see, e.g. [49, Ch 3] for definition). Alternatively, the computation could loop forever or terminate in a reject state. A Turing-Recognizable language is one for which there exists a Turing Machine that accepts only the strings in the language, and either rejects or does not halt at other strings. A set of languages $S$ is nontrivial if there exists a Turing-Recognizable language that it contains, and a different Turing-Recognizable language that it does not contain.
Any I/O relationship for $C$ can be reduced to a decision problem/language (i.e., a mapping from finite string input to binary “accept/reject” output) by designating one of its possible outputs as “reject”, and accepting strings with any other output. Thus, an I/O relationship for $C$ can be viewed as a language of $C$. Thus, our definition of I/O relationship sets $F_p$ naturally extends to nontrivial $F_p$’s. We now state Rice’s theorem (Theorem 2), which provides an undecidability result that we rely on to derive our undecidability result (Theorem 3) by connecting our class of $C$’s with Rice’s theorem. While originally proven in [28], for simplicity, we use the theorem statement in [47].

**Theorem 2** (Rice’s theorem [28], [47]). Let $S$ be a nontrivial set of languages. It is undecidable whether the language recognized by an arbitrary Turing machine lies in $S$.

**Theorem 3.** For an $H$ containing a nontrivial $F_p$, for any $L \geq 0$, $M = \infty$, and $Q = 0$, there is no Turing machine $A$ which can accept as input, an arbitrary computational system $C$ with infinite set $S$ and computable functions evaluated at nodes, and output $TM_{C,M,H,Q}$ that satisfies the reverse engineering properties in Definition 5.

**Proof.** Assuming there were such a Turing machine $A$, we construct a Turing machine $M$ (that will solve Rice’s problem) as follows: accept input string $s$ encoding Turing machine corresponding to $C$ (Theorem 1 states that, with infinite $S$ and computable functions, the class of $C$’s is Turing equivalent), and give $s$ as input to $A$. If $A$ outputs a Turing Machine that, on input $p$ (for a nontrivial $F_p$), outputs ‘no solution’ or $> 0$ interventions, then $M$ outputs 0, else (i.e., for 0 interventions) it outputs 1. Then $M$ decides whether an input Turing machine has language in $F_p$, contradicting Theorem 2. □

V. SOME EXAMPLES OF REVERSE ENGINEERED SYSTEMS

- a) Line network in a grid
- b) Network coding butterfly (with $S$ and $O$ nodes)
- c) FFT butterfly

![Fig. 1. Examples C’s considered in Section V. Input nodes have incoming arrows (with no source), output nodes have outgoing arrows (with no destination). In a), an alternative destination node is shown in red, and blue nodes show where interventions need to be performed to change the I/O relationship to have the output go out of this red node. b) is the classic network-coding butterfly, where I/O relationships need to be changed to affect a specific output node. c) is the classic FFT butterfly network (with $S$ and $O$ nodes) in doing so. The key observation is that (see Figure 1c) if an I/O change inside a $F_p$ can arise from interventions on a single node, then one such node is the one that we arrive at by stepping leftwards (by $\log_2(B)$ steps if $B$, the number of affected output nodes, is $2^k$ for some $k \in N$) from any of the affected output nodes (see Fig. 1c for intuition). The TM output by $A$ executes the following: the input $p$ provides the indices of the output nodes affected by the intervention. If the number of these nodes is not $2^k$ for some $k \in N$, output “no solution” (a single intervention is insufficient). If it is, then choose the first such output node, and, looking at the FFT architecture, traverse left by $k$ steps. Ask the oracle if an intervention on this node can produce a desired I/O pattern. If yes, then a solution is this node. If not, output “no solution” (> 1 interventions needed).

REFERENCES

[1] Praveen Venkatesh, Sanghamitra Dutta, Neil Ashim Mehta, and Pulkit Grover. Can information flows suggest targets for interventions in neural circuits? Advances in Neural Information Processing Systems (NeurIPS), 2021.
[2] Neil A Mehta and Pulkit Grover. Efficient interventions in a neural circuit from observations: An information-theoretic study. submitted, 2022.

[3] Yuri Lazebnik. Can a biologist fix a radio?—or, what I learned while studying apoptosis. Cancer cell, 9(2):151–202, 2002.

[4] Ian H Stevenson and Konrad P Kording. How advances in neural recording affect data analysis. Nature neuroscience, 14(2):139–142, 2011.

[5] Eric Jonas and Konrad Paul Kording. Could a neuroscientist understand a microprocessor? PLoS computational biology, 13(1):e1005268, 2017.

[6] Peiran Gao and Surya Ganguli. On simplicity and complexity in the brave new world of large-scale neuroscience. Current opinion in neurobiology, 32:148–155, 2015.

[7] John J Hopfield and Carlos D Brody. What is a moment? Transient synchrony as a collective mechanism for spatiotemporal integration. Proceedings of the National Academy of Sciences, 98(3):1282–1287, 2001.

[8] W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. Proceedings of the National Academy of Sciences, 116(44):22071–22080, 2019.

[9] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08668, 2017.

[10] Alexander R Galloway and Jason R LaRivière. Compression in philosophy. boundary 2, 44(1):125–147, 2017.

[11] Carl F Craver. When mechanistic models explain. Synthese, 153(3):355–376, 2006.

[12] GJ Chaitin. Epistemology as information theory: From leibniz to omega. arXiv preprint math/0506552, 2005.

[13] Philippe Bracke, Anupam Datta, Carsten Jung, and Shayak Sen. Machine learning explainability in finance: an application to default risk analysis. Bank of England Working Paper, 2019.

[14] James Woodward. Scientific explanation. Stanford Encyclopedia of Philosophy, 2011.

[15] Dongjin Seo, Ryan M Neely, Konlin Shen, Ultskar Singhal, Eldad Alon, Jan M Rahaye, Jose M Carmena, and Michel M Maharbiz. Wireless recording in the peripheral nervous system with ultrasonic neural dust. Neuron, 91(3):529–539, 2016.

[16] James K Trevathan, Ian W Baumgart, Evan N Nicolai, Brian A Gosink, Christine Grubert, Barbara Hadrysiewicz, Nikolai Axmacher, Matthias Lemke, David W Cook-Mardis, Michael X Cohen, et al. Nucleus accumbens deep brain stimulation decreases ratings of depression and anxiety in treatment-resistant depression. Biological psychiatry, 67(2):110–116, 2010.

[17] Patric Blomstedt, Richard L Sjöberg, Maja Hansson, Owe Bodlund, and Mats Forssell. Chattayna Goswami, Ashwati Krishnan, Mayasreema Panzar and, Pulkit Grover. Effect of skull thickness and conductivity on current propagation for noninvasively injected currents. Journal of Neural Engineering, 18(4):046042, 2021.

[18] Bettina H Bewernick, René Hurlemann, Andreas Matusch, Sarah Kayser, Bettina H Bewernick, René Hurlemann, Andreas Matusch, Sarah Kayser, Christiane Grubert, Barbara Hadrysiewicz, Nikolai Axmacher, Matthias Lemke, David W Cook-Mardis, Michael X Cohen, et al. Nucleus accumbens deep brain stimulation decreases ratings of depression and anxiety in treatment-resistant depression. Biological psychiatry, 67(2):110–116, 2010.

[19] Michael Y Oh, David B Cohen, and Donald M Whiting. Deep brain stimulation in the treatment of obsessive-compulsive disorder: World neurosurgery, 80(6):e245–e253, 2013.

[20] Christoph Molnar. Interpretable machine learning: https://christophm.github.io/interpretable-ml-book/, 2020.

[21] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In International Conference on Machine Learning, pages 3319–3328. PMLR, 2017.

[22] Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In Advances in Neural Information Processing Systems, volume 30, pages 6970–6979. Curran Associates, Inc., 2017.

[23] R Christopher Pierce and Fair M Vassoler. Deep brain stimulation for the treatment of addiction: basic and clinical studies and potential mechanisms of action. Psychopharmacology, 229(3):487–491, 2013.

[24] Michael Sipser. The fast fourier transform in a finite field. Journal of Algorithms, 6(2):170–187, 1985.

[25] Karl Kumbier, Brian A Gosink, and Michael Maharbiz. Wireless recording in the peripheral nervous system with ultrasonic neural dust. Neuron, 91(3):529–539, 2016.

[26] Henry Gordon Rice. Classes of recursively enumerable sets and their decision problems. Transactions of the American Mathematical Society, 74(2):358–366, 1953.

[27] Pavan Venkatesh, Sanghamitra Dutta, and Pulkit Grover. Information flow in computational systems. IEEE Transactions on Information Theory, 66(9):5456–5491, 2020.

[28] Henri A Fourtine. Functional and effective connectivity: a review. Brain connectivity, 1(1):13–36, 2011.

[29] Pavan Venkatesh, Sanghamitra Dutta, and Pulkit Grover. How should we define information flow in neural circuits? In 2019 IEEE International Symposium on Information Theory (ISIT), pages 176–180, 2019.

[30] N Tishby. The information bottleneck method. In Proc. 37th Annual Allerton Conference on Communications, Control and Computing, 1999. Allerton Conference, pages 368–377, 1999.

[31] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. Advances in Neural Information Processing Systems, 33:2958–2969, 2020.

[32] Timothy K Horváth, Brooks Bishofberger, and Christof Koch. An analog VSLI saccadic eye movement system. In Proceedings of the 6th International Conference on Neural Information Processing Systems, pages 582–589, 1993.

[33] Theodore W Berger, Robert E Hampson, Dong Song, Anushka Goonawarden, Vasili Z Marmarelis, and Sam A Deadwyler. A cortical neural prostheses for restoring and enhancing memory. Journal of neural engineering, 8(4):046017, 2011.

[34] Philippo Karlo Dostilović, Mario Brčić, and Nikica Huplić. Explainable artificial intelligence: A survey. In 2018 41st Annual international conference on information and communication technology, electronics and microelectronics (MIPRO), pages 0210–0215, 2018.

[35] David Marr. Vision. MIT Press, Cambridge MA, 1982.

[36] D Peebles and RP Cooper. Thirty years after Marr’s vision: Levels of analysis in cognitive science. Topics in Cognitive Science, 7(2):187–190, 2015.

[37] Benjamin James Lansdell and Konrad Paul Kording. Towards learning-to-learn. Current Opinion in Behavioral Sciences, 29:45–50, 2019.

[38] Zbigniew Szymanski. The fast fourier transform in a finite field. Journal of Algorithms, 9(2):270–291, 1988.

[39] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. arXiv preprint arXiv:1410.5401, 2014.

[40] R. Ahlswede, Ning Cai, S. Y. R. Li, and R. W. Yeung. Network information flow. IEEE Trans. Inf. Th., 46(4):1204–1216, July 2000.

[41] Clark David Thompson. A Complexity Theory for VLSI. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1980. AAI8100621.

[42] Herbert A Simon. Boundary rationality. In Utility and probability, pages 15–18. Springer, 1990.

[43] Christoph A Simons. Rational inattention: a research agenda. Technical report, Discussion Paper Series 1, 2005.

[44] Hans Hüttel. Rice’s theorem: http://people.cs.aau.dk/~srba/courses/cc-08/riuce-uk.pdf, 2007.

[45] Bava T Siegelmann and Eduardo D Sontag. On the computational power of neural nets. Journal of computer and system sciences, 50(1):132–150, 1995.

[46] Michael Sipser. Introduction to the Theory of Computation. Course Technology, Boston, MA, third edition, 2013.

[47] John M Pollard. The fast fourier transform in a finite field. Mathematics of computation, 25(114):365–374, 1971.

[48] Scott J Russo and Eric J Nestler. The brain reward circuitry in mood disorders. Nature Reviews Neuroscience, 14(9):609–625, 2013.

[49] Praneeth Namburi, Ream Al-Hasani, Gwendolyn G Calhoon, Michael R Nicosia, and Marwan I Hariz. Deep brain stimulation in the treatment of obsessive-compulsive disorder: World neurosurgery, 80(6):e245–e253, 2013.
APPENDIX A
PROOF OF THEOREM 1.1

We denote a Deterministic Finite Automaton by DFA $(\Sigma, Q_{dfa}, s_0, \delta, F)$ consisting of a finite set of states $Q_{dfa}$, a finite set of input symbols called the alphabet $\Sigma$, a transition function $\delta : Q_{dfa} \times \Sigma \rightarrow Q_{dfa}$, an initial state $s_0 \in Q_{dfa}$, and a set of accepting states $F \subseteq Q_{dfa}$. To simulate the DFA $(\Sigma, Q_{dfa}, s_0, \delta, F)$, we construct a $\mathcal{C}$ as follows:

1) Nodes of the graph $G$ in $\mathcal{C}$ are the states of the DFA, with an additional output node $\{o\}$, i.e., $\mathcal{V} = Q_{dfa} \cup \{o\}$.

2) Edges of $G$ are: (i) all the transition edges of the DFA, i.e., for every two states $s, t \in Q_{dfa}$ for which there is $d \in \Sigma$ such that $\delta(s, d) = t$, there is an edge $s \rightarrow t$ in $G$; (ii) self-loops at every node (if not defined by (i)); and (iii) For each accepting state of the DFA ($s \in F$), an edge $s \rightarrow o$.

3) $S = \Sigma \cup \{\text{start}, \text{fin}, \text{blank}\}$, i.e., for defining $S$ in $\mathcal{C}$, we use $\Sigma$, and, additionally, start, fin (finish), and blank symbols.

4) Each node receives the computational input, $d$, at each time step.

5) Initialize all states of nodes of $\mathcal{C}$, except the node corresponding to $s_0$, with blank, and the state of the node corresponding to $s_0$ with start. The function computed at each node $s \in \mathcal{V} \setminus \{o\}$, on the transmissions it receives (say $x_1, \ldots, x_k$; exactly one of the $x_j$’s is not blank) and the computation input $d$ is

$$f_s(\text{‘blank’}, \ldots, x_j, \ldots, \text{‘blank’}, d) = \begin{cases} d, & \text{if } x_j \in \Sigma, \delta(s_j, x_j) = s \\ d, & \text{if } x_j = \text{‘start’}, s = s_0 \\ \text{‘blank’} \text{ else,} & \end{cases}$$

and the output node computes:

$$f_o(\text{‘blank’}, \ldots, x_j = \text{‘fin’}, \ldots, \text{‘blank’}, d) = \begin{cases} 1, & \text{if } s_j \in F \\ 0, & \text{else.} \end{cases}$$

With this construction, the output node outputs 1 on a computational input string iff the DFA accepts the string.

APPENDIX B
ALTERNATIVE DEFINITIONS

Here, we introduce two non-interventional definitions of reverse engineering, and show that those are also undecidable.

Definition 7 (Single-Node RE). An agent $A$ is said to Single-Node Reverse Engineer a computational system $C$ if given any node $i$ of $C$, it can determine whether there is any input to the computation system such that at some time instant, the node $i$ stores a non-zero value (i.e., whether the node is ever activated).

Theorem 4. There is no Turing machine $A$ which can accept as input, an arbitrary computational system $C$ (having countably infinite alphabet) and arbitrary node $i$ of $C$, and output whether the node $i$ is ever activated.

Proof. Suppose there were such a Turing machine $A$. Then, we can construct a Turing machine $M_E$ that decides the language $E_{TM} = \{\langle M \rangle : M \text{ is a TM and } L(M) = \emptyset\}$ as follows: $M_E$ accepts input string encoding Turing machine $(M)$, creates an encoding of the corresponding computation system $C_M$ whose output node is labelled as $v$. $M_E$ simulates $A$ on input $\langle C_M, v \rangle$ and outputs true iff $A$ outputs true.

Then $M_E$ as described above decides $E_{TM}$ since node $v$ of the constructed computation system $C_M$ is ever activated iff $M$ ever accepts an input string. However we know that $E_{TM}$ is undecidable (Theorem 5.2, [49]), thus such a Turing machine $M_E$ cannot exist.

This previous result shows that determining if a node in a neural circuit even represents a message of interest is undecidable. The result that follows this next definition shows that even estimating approximations of functions being computed (I/O relationships) can be undecidable.

Definition 8 ($(k,f)$-Approximate RE). Given a computable function $f : \mathbb{Q} \rightarrow \mathbb{Q}$ and number $k \in \mathbb{N}$, an agent $A$ is said to $(k,f)$-Reverse Engineer a computational system $C$, if it can determine whether $C$ computes a $k$-approximation of $f$, i.e., whether on every input string $x \in \mathbb{Q}$, we have

$$\frac{1}{k} |f(x)| \leq |C(x)| \leq kf(x)$$

Theorem 5. For every computable function $f$, there is no Turing machine $A$ which can accept as input an arbitrary computation system $C$ and output whether $C$ computes a $k$-approximation of $f$.  

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Proof. As in the previous theorem, suppose there were such a Turing machine $A$. Then, we construct a Turing machine $M_E$ deciding $T_{TM}$ as follows: $M_E$ accepts input string encoding Turing machine $\langle M \rangle$. It constructs an encoding $\langle C_M \rangle$ of a computation system which takes input string $x$, first simulates computing $M(x)$. Then, if $M$ accepts $x$, outputs $k f(x) + 1$, else outputs $f(x)$. Then $M_E$ simulates $A$ on input $\langle C_M \rangle$ and outputs true iff $A$ determines that $C$ is a $k$-approximation of $f$.

Thus, $M_E$ described as above decides $T_{TM}$ since the constructed $C_M$ computes a $k$-approximation of $f$ iff $M$ rejects all inputs. However, as we know, $T_{TM}$ is undecidable. Thus by contradiction, such an $A$ does not exist. 

\section*{Appendix C}

\textbf{Examining the state of understanding of an exemplar brain network: the reward network in the brain}

The brain’s reward network is a complex circuit that is responsible for desire for a reward, positive reinforcement, arousal, etc. Dysfunction in this network can result in depression, OCD, addiction, etc. The reward network consists of several large centers, such as the ventral tegmental area (VTA), the Amygdala (Amy), the Nucleus Accumbens (NAc), the Hippocampus (Hipp), the Prefrontal Cortex (PFC), and the Orbitofrontal Cortex (OFC), that interact with one another in complex ways. A simplified version of this network is illustrated in Fig. 2.

Decades of scientific research has helped develop some understanding of how these large brain regions interact. Below, we provide a brief overview of this body of work in the context of representation of “valence” (positive or negative emotion) in the reward network. We refer biologically-inclined readers to [51], [52] as starting points for a deeper study. We provide an overview of the understanding of the reward network, and how it can suggest strategies for interventions. We want the reader to observe that, while the understanding is quite detailed and includes causal effects, it is still far from the reverse-engineering goal laid out in our work. This gap could help set an aim for, and/or the ability to intervene on, this circuit (e.g., if some nodes are inaccessible for stimulation, or less explored for their functional understanding).

Back in 1939, Klüver and Bucy [53] observed (in monkeys) that lesioning in the temporal lobe and amygdala led to extreme emotional changes, including loss of fear responses, failure to learn from aversive stimuli, and increased sexual behavior (leading to what is called Klüver-Bucy syndrome in humans with similar injuries). Since this work, animal studies, including in mice, rats, monkeys, etc., have been frequently used to understand how the brain responds to rewarding/pleasant (positively valenced) or aversive (negatively valenced) stimulus presentation. Many studies have since examined which regions of the brain “represent valence”, in that their neural response statistics change when positive vs negatively valenced stimuli are presented. These studies show that many (broad) regions represent valence, including the amygdala [54], [55], nucleus accumbens [56], ventral tegmental area [57], orbitofrontal and prefrontal cortex [58], lateral hypothalamus [59], subthalamic nucleus [60], hippocampus [54], etc. (see [52] for an excellent survey). Recently, advances in neuroengineering, especially in optogenetics [61] and minimally invasive implants [15], enable finer-grained examination within these broad brain regions, including spatiotemporally precise interventions, examining neural “populations”, i.e., collections of neurons within the same broad region that are similar “functionally” (i.e., in how they respond to rewarding or aversive stimuli), genetically (e.g., in the type of neurons), and/or in their connectivity (which region they connect with). For Nucleus Accumbens, for instance, these techniques have led to further separation of the region into its core vs its shell. Dopamine release in the core (often due to activation of the VTA by a rewarding or aversive stimulus) appears to reinforce rewarding behavior, while the same dopamine released in the shell can lead to both rewarding and aversive stimuli. E.g. an addiction ‘hotspot’ is found in the medial shell, while in another location, a ‘coldspot’ reduces response to addictive stimuli, suggesting a fine control by the two populations (see [52]). Similarly refined understanding has been developed for other nodes, e.g. the amygdala and the VTA (see [52]).

Thus, at first glance, one way think that that estimation of what stimulus presentation affects which neural population, and how interventions on a neural population affect processing of a stimulus, are increasingly at a spatial resolution that is required to answer reverse engineering questions we pose here (they will only be further enabled by recent advances in neuroengineering [61], [15]). Indeed, many clinicians are already utilizing this understanding to do surgical implants that intervene on functioning of this network, including for depression [18], OCD [19], addiction [20], obesity [21], etc., when the disorder is extreme. However, our understanding of the network is still severely lacking: we do not know, for instance, what the functions computed at these nodes are, which can have a significant effect on what the minimal intervention is.

These limitations in understanding of this network affects our ability to provide optimized solutions (e.g. those that are minimal in the sense discussed in our paper). This might seem intuitive, but for completeness we include a simple example of the influence of the Nucleus Accumbens on subregions nodes (PFC and OFC). E.g., suppose its output to PFC, $Y_{\text{NAc} \rightarrow \text{PFC}} = I_{\text{HS}} - I_{\text{CS}}$ is the difference of the outputs of the hotspot $I_{\text{HS}}$ and the coldspot $I_{\text{CS}}$ discussed above. Further, the output to OFC could be A) the ratio; or B) the difference of the outputs of the hotspot and the coldspot. That is, $Y_{\text{NAc} \rightarrow \text{OFC}}^{(1)} = \frac{I_{\text{HS}} - I_{\text{CS}}}{I_{\text{CS}}}$ and $Y_{\text{NAc} \rightarrow \text{OFC}}^{(2)} = I_{\text{HS}} - I_{\text{CS}}$. The goal is to produce an
intervention that makes $Y_{\text{NAc-OFC}} = Y_{\text{NAc-PFC}} = (1 - \gamma)I_{\text{HS}}$ (i.e., $\mathcal{H}$ is constituted by the I/O relationships of this form for NAc, one for each $\gamma$). Now, let’s assume that links (arising from separate nodes) from the hotspot and the coldspot populations go to PFC and OFC, but the coldspot receives $I_{\text{CS}}$ from a common ancestor. It is easy to see that in this case, the reverse engineering solution depends on which is the actual function: if $Y_{\text{NAc-PFC}} = Y_{\text{NAc-OFC}} = I_{\text{HS}} - I_{\text{CS}}$, intervening on the coldspot’s ancestor will suffice (namely, by setting $I_{\text{CS}} = \gamma I_{\text{HS}}$). However, if $Y_{\text{NAc-OFC}} = \gamma I_{\text{HS}} - I_{\text{CS}}$, the set of minimal interventions should be of cardinality two, constituted by interventions on two locations within the coldspot, to get both outputs to equal $(1 - \gamma)I_{\text{HS}}$ (namely, one that outputs to PFC should have the signal $\gamma I_{\text{HS}}$, whereas one that outputs to OFC should have the signal $1/(1 - \gamma)$). Observe that the qualitative relationship between how $I_{\text{HS}}$ and $I_{\text{CS}}$ affect the outputs is similar in the two possibilities considered here (i.e., the first increases the outputs, and the second reduces it).

We think that this suggests the possibility of subsequent work which uses a computer-scientific and information-theoretic lens to contribute to design of experiments (observational and interventional) for garnering the needed inferences about this computational system (such as modeling functions computed at nodes, not just activation/influence of a node).

**APPENDIX D**

**DISCUSSION AND LIMITATIONS**

What aspect in our work makes it motivated by neuroscience? After all, our computation system model is fairly general, and builds on prior work in theoretical computer science (see, for instance, work on “VLSI theory” in the 1980s). While, intellectually, finding a set of minimal interventions demonstrates strong understanding of how a computational system works, we believe that, operationally, the minimal intervention aspect is most closely tied to networks in neuroscience. Intervening on machine-learning networks (such as ANNs), can we find no natural reason why one should attempt to find minimal interventions. Editing few nodes and/or edges of ANNs implemented in hardware is not a problem that is relevant in today’s implementations. However, this problem arises naturally in neuroscience, as one would naturally want to intervene on as few locations as possible (say, because each location requires a surgical intervention). Of relevance here is a recent work on cutting links in epileptic networks, where the authors seek a similar minimalism [62].

Our **definition of what constitutes a minimal intervention** could be tied more closely to biological constraints and peculiarities. While our definition is motivated from recent surgical interventions on the reward circuitry and advances in neuroscience, sometimes, a noninvasive intervention, even if more diffused, might be preferred to an invasive intervention because it does not require implantation (implantation has risk of infection, need for removal etc.). Similarly, it is known that in the brain (even in the reward network [52]), different populations have different likelihood of having neurons that represent and affect valence, and different neurons also have different magnitudes of effects they produce on the network’s reward valuation. The practical difficulties of finding a neuron close to where an implant is placed, and/or difficulty-levels of surgical interventions, might need to be incorporated in our model.

As a practical direction, we think that clinical neuroscience research should not only focus on describing the system or examining some causal pathways of interventions, but also actively on modifications and interventions at the fewest possible locations (or minimal in ways suited to the specific disorder) that can change the I/O behavior to a desirable one. It is conceivable that a neuroscientist might want to demonstrate how they are able to “control” the circuit as a way of certifying their understanding of the system. From this perspective, we recognize that this demonstration of control (to any I/O behavior) of the circuit is stronger than what might be needed for getting a **specific** behavior that is desirable, and this can be captured in our definition by careful choice of $\mathcal{H}$.

Our nodes-and-edges discrete-time model is a crude one, because even single cells can exhibit extremely complex dynamics [63], [64]. However, models such as ours are commonly used (e.g. [29], [65], [66] and references therein) in computational neuroscience as a first step, and have been applied to real data. Here, our goal is to use these models to formally state the reverse engineering definition, which allows us to illustrate how reverse engineering could be achieved, and obtain undecidability results for a class of problems.

On our requirements, one can replace bounded memory constraints to other constraints [44] (e.g., computational or informational [46]), or also seek **approximately** minimal interventions. We believe that (based on simplicity of results in Appendix B) the general problem will continue to be undecidable for many such variations. Hardness/impossibility results have continued to inform and refine approaches across many fields (e.g. hardness of Nash equilibrium [67] and of decentralized control problems [68], and recent undecidability results in physics [69], [70], among others). An undeniable consequence of our result is that there cannot exist an algorithm that solves the reverse engineering problem posed here in general. There exist cases that are extremely hard to reverse engineer, even if (as illustrated by our examples in Section V), in many cases, reverse engineering can be accomplished.

On our undecidability result, note that if the alphabet of computation is finite, then the reverse engineering problems posed here are decidable. However, in that case, the model for brain is also not Turing complete. Finally, one must note that undecidability is not an artifact of our definition. As shown in Appendix B, other plausible definitions we considered also yielded analogous undecidability results. Our proof technique extends to many related definitions, as illustrated by the relaxed assumptions under which we are able to prove the results (and, indeed, the relaxed assumptions under which Rice’s theorem is obtained). As rapid advances in neuroengineering enable breakthrough neuroscience, challenging conceptual and mathematical problems will arise. In fact, today, both AI and neuroscience are using increasingly complex models and are asking increasingly complex interpretability/reverse engineering questions. It is worth asking whether instances of this question are undecidable, and, if decidable, how the complexity of a reverse engineering problem scales with the problem size.