Natural Computing
https://doi.org/10.1007/s11047-020-09805-3

The representational entity in physical computing

Susan Stepney[1] · Viv Kendon[2]

Accepted: 1 September 2020

Abstract
We have developed abstraction/representation (AR) theory to answer the question "When does a physical system compute?" AR theory requires the existence of a *representational entity* (RE), but the vanilla theory does not explicitly include the RE in its definition of physical computing.Here we extend the theory by showing how the RE forms a linked complementary model to the physical computing model. We show that the RE does not need to be a human brain, by demonstrating its use in the case of intrinsic computing in a non-human RE: a bacterium.

Keywords Unconventional computing · Bacterial computing · Representation

1 Introduction

Many and diverse physical substrates are proposed for unconventional computing, from relativistic and quantum systems to chemical reactions and slime moulds, from carbon nanotubes to non-linear optical reservoir systems, from amorphous substrates to highly engineered devices, from general purpose analogue computers to one-shot devices.In another domain, biological systems are often said to perform information processing. In all these cases it is crucial to be able to determine when such substrates and systems are specifically computing, as opposed to merely undergoing the physical processes of that substrate.

In order to address this question, we have been developing *abstraction/representation* theory (AR theory). This is a framework in which science, engineering/technology, computing, and communication/signalling are all defined as a form of *representational activity*, requiring the fundamental use of the representation relation linking physical system and abstract model in order to define their operation (Horsman et al. 2014; Horsman 2015). Within this framework, it is possible to distinguish scientific experimentation on a novel substrate (an activity necessary to characterise the computational capabilities of a substrate) from the performance of computation by that substrate. This is needed to distinguish cases where a substrate superficially appears to be computing, because it sometimes produces a state that resembles a computational result (which can be determined only by comparison with a separately computed result), from cases where a substrate is reliably and consistently producing desired computational results.

In work following on from the original definitions, Horsman et al. (2017b) provide a high level overview, Horsman et al. (2018) delve into more philosophical aspects, and Horsman et al. (2017a) present an example of intrinsic computation: signalling in bacteria. Also see Horsman et al. (2014, 2018) for references to the wider unconventional computing and philosophical literature.

AR theory requires the existence of a representational entity (RE) to support the representation relation. One issue glossed over in our previous descriptions of AR theory that becomes crucial when analysing computation in systems where the RE is not a human or conscious user, is the relationship between the physical RE and the physical computer. Here we enrich AR theory by incorporating the RE explicitly, and showing how it relates to the physical computing process.

The structure of the paper is as follows. In Sect. 2 we summarise the current formulation of AR theory. In Sect. 3 we extend the theory to include the RE explicitly. In Sect. 4

This is an extended journal version of Stepney and Kendon (2019), a UCNC 2019 conference paper, including new examples of mental arithmetic and of extrinsic bacterial computing, a new discussion of various modelling issues, and several clarifications throughout.

✉ Susan Stepney
susan.stepney@york.ac.uk

[1] Department of Computer Science, University of York, York, UK

[2] Department of Physics, Durham University, Durham, UK

Published online: 18 September 2020

we demonstrate how the extended theory allows us to capture and model intrinsic computing, including an example in a bacterium.

2 AR theory in a nutshell

2.1 Our view of physical computing

AR theory has been developed to answer the specific question of when a physical system is computing (Horsman et al. 2014). The answer hinges on the relationship between an abstract object (a computation) and a physical object (a computer). It employs a language of relations, not from mathematical objects to mathematical objects (as is usual in mathematics and theoretical computer science), but between physical objects and those in the abstract domain. The core of AR theory is the *representation relation*, mapping from physical objects to abstract objects. Experimental science, engineering, and computing all require the interplay of abstract and physical objects via representation in such a way that their descriptive diagrams *commute* such that the same result can be gained through either physical or abstract evolutions (see Sect. 2.3). From this, Horsman et al. (2014) define computing as *the use of a physical system to predict the outcome of an abstract evolution*.

2.2 Representation

AR theory has physical objects in the domain of material systems, abstract objects (including mathematical and logical entities), and the representation relation that mediates between the two. The distinction between the two spaces, abstract and physical, is fundamental in the theory, as is their connection *only* by the (directed) representation relation. An intuitive example is given in Fig. 1: a physical switch is *represented* by an abstract bit, which in this case takes the value 0 for switch state up, and 1 for switch state down. Note, however, that AR theory is not a dualist theory in the sense of Descartes. Everything in the theory is physical in some form. The symbols in the *Abstract* domain in Fig. 1 are instantiated as ink on paper or pixels on the screen as you read this. What makes them *abstract* in AR theory is that this physical form is to some degree arbitrary, and can change, while still corresponding to the same abstract object.

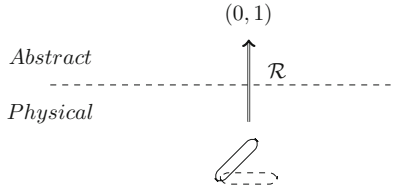

Fig. 1 Basic representation has three components: the space of physical objects (here, a switch with two settings); the space of abstract objects (here, a binary digit); the directed representation relation $\mathcal{R}$ mediating between the spaces

An example of a physical object in the domain of material entities is a *computer*. It has, usually, internal degrees of freedom, and a physical time evolution that transforms initial input to final output states. An example of an abstract object is a *computation*, which is a set of objects and relations as described in one of the logical formalisms of theoretical computer science. Likewise, an object such as a bacterium is a physical entity, and its theoretical representation within biology is an object in the domain of abstract entities.

The central role of representation leads to the requirement for a *representational entity* (RE). The RE supports the representation relation between physical and abstract. AR theory does not require the RE to be human, or conscious; see Horsman et al. (2017a) for an example of a bacterial RE, which is expanded on in Sect. 4.2 here.

The elementary *representation relation* is the directed map from physical objects to abstract objects, $\mathcal{R}_\mathcal{T} : \mathbf{P} \to M$, where $\mathbf{P}$ is the set of physical objects, and M is the set of abstract objects. We subscript the relation $\mathcal{R}$ with a theory $\mathcal{T}$ to indicate that the relation is theory-dependent. When a physical object $\mathbf{p}$ and an abstract object $m_\mathbf{p}$ are connected by $\mathcal{R}_\mathcal{T}$ we write them as $\mathbf{p} \mapsto m_\mathbf{p}$. The abstract object is then said to be the *abstract representation* (under the given theory) of the physical object. This basic representation is shown in Fig. 2a.

Abstract evolution takes abstract objects to abstract objects, which we write as $C_\mathcal{T} : M \to M$. Again, we subscript with theory $\mathcal{T}$ to indicate that C is theory-dependent. An individual example is shown in Fig. 2b, for the mapping $C_\mathcal{T}(m_\mathbf{p})$ taking $m_\mathbf{p} \mapsto m'_\mathbf{p}$. The corresponding physical evolution map is given by $\mathbf{H} : \mathbf{P} \to \mathbf{P}$. For individual elements in figure 2c this is $\mathbf{H}(\mathbf{p})$ which takes $\mathbf{p} \mapsto \mathbf{p}'$.

2.3 ε-commuting diagrams

In order to link the final abstract and physical objects, we apply the representation relation to the outcome state of the physical evolution, to give its abstract representation $m_{\mathbf{p}'}$, Fig. 2d. We now have two abstract objects: $m'_\mathbf{p}$, the result of the abstract evolution, and $m_{\mathbf{p}'}$, the representation of the result of the physical evolution. For some (problem-dependent) error quantity ε and distance function $d()$, if $d(m_{\mathbf{p}'}, m'_\mathbf{p}) \leq \varepsilon$ (or, more briefly, $m_{\mathbf{p}'} =_\varepsilon m'_\mathbf{p}$), then we say that the diagram 2(d) *ε-commutes*.

Commuting diagrams are fundamental to the use of AR theory. If the relevant abstract and physical objects form an ε-commuting diagram under representation, then $m_\mathbf{p}$ is a

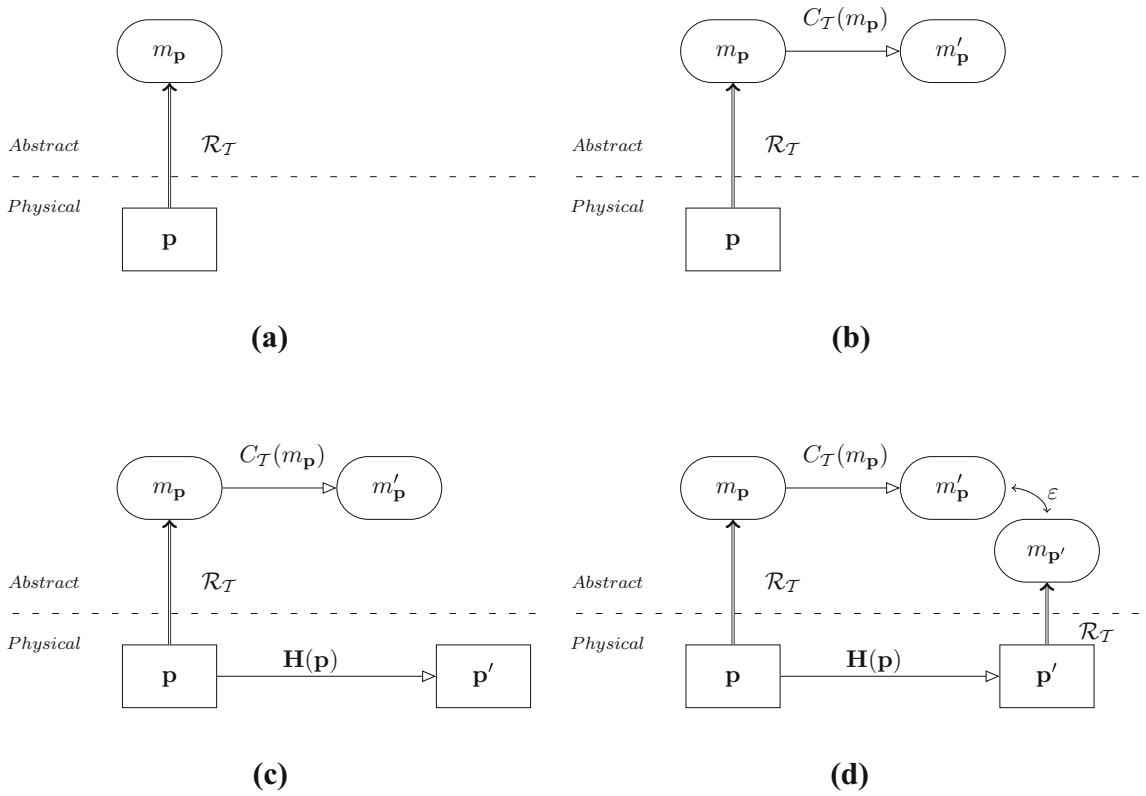

Fig. 2 Parallel evolution of an abstract object and the physical system it represents. **a** The basic representation: physical system $\mathbf{p}$ is represented abstractly by $m_{\mathbf{p}}$ using the modelling representation relation $\mathcal{R}_{\mathcal{T}}$ of theory $\mathcal{T}$. **b** Abstract dynamics $C_{\mathcal{T}}(m_{\mathbf{p}})$ give the evolved abstract state $m'_{\mathbf{p}}$. **c** Physical dynamics $\mathbf{H}(\mathbf{p})$ give the final physical state $\mathbf{p}'$. **d** $\mathcal{R}_{\mathcal{T}}$ is used again to represent $\mathbf{p}'$ as the abstract output $m_{\mathbf{p}'}$. If $m_{\mathbf{p}} =_{\varepsilon} m_{\mathbf{p}'}$, the diagram ε-commutes. (Adapted from Horsman et al. (2014).)

faithful abstract representation (up to ε) of physical system $\mathbf{p}$ for the evolutions $C_{\mathcal{T}}(m_{\mathbf{p}})$ and $\mathbf{H}(\mathbf{p})$.

The existence of such ε-commuting diagrams define what is meant by a faithful abstract representation of a physical system. The final state of a physical object undergoing time evolution can be known *either* by tracking the physical evolution and then representing the output abstractly, *or* by evolving the abstract representation of the system; and the two results differ by less than the problem-dependent ε. In the first case, the 'lower path' of the diagram is followed; in the latter, the 'upper path'.

Finding out which diagrams ε-commute is the business of basic experimental science; once commuting diagrams have been established they can be exploited through engineering and technology.

2.4 Compute cycle

Figure 2d shows the basic 'science cycle', of representing a physical system, and determining whether $C_{\mathcal{T}}$ is a sufficiently good abstract model of its behaviour, by requiring that $m_{\mathbf{p}} =_{\varepsilon} m_{\mathbf{p}'}$ for sufficiently many different initial states $\mathbf{p}$ to have confidence in $C_{\mathcal{T}}$ and $\mathcal{R}_{\mathcal{T}}$. There are derived variants of this diagram that capture the 'engineering cycle', and the related 'compute' cycle. See the original references cited in Sect. 1 for details; here we focus on the compute cycle.

An ε-commuting diagram in the context of *computation* also connects the physical computing device, $\mathbf{p}$, and its abstract representation $m_{\mathbf{p}}$. But to do so it makes use of the *instantiation* relation $\widetilde{\mathcal{R}}_{\mathcal{T}} : M \rightarrow \mathbf{P}$. Here, instead of saying abstract object $m_{\mathbf{p}}$ represents physical system $\mathbf{p}$, we say that physical system $\mathbf{p}$ instantiates abstract object $m_{\mathbf{p}}$. Whereas the representation relation is primitive, the instantiation relation is a derived relation, based on multiple science cycles, abbreviated as $\widetilde{\mathcal{R}}_{\mathcal{T}}$; see original references for full details.

The use of $\widetilde{\mathcal{R}}_{\mathcal{T}}$ acknowledges that a computer is physical system engineered (or possibly evolved) to have a particular behaviour, rather than a natural physical system being scientifically modelled. The full compute cycle is shown in Fig. 3, starting from initial abstract problem, through instantiation into a physical computer, physical evolution of the device, followed by representation of the final physical state as the abstract answer the the problem.

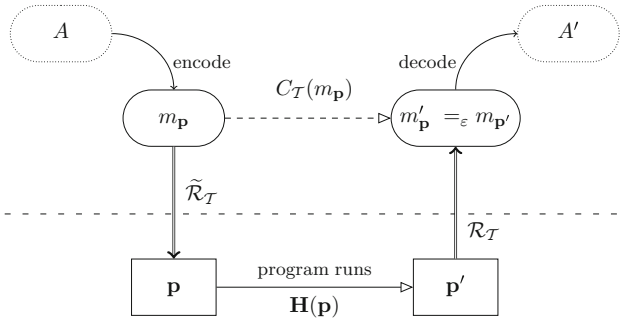

Fig. 3 Physical computing in AR theory. An abstract problem A is encoded into the model $m_{\mathbf{p}}$; the model is instantiated into the physical computer state $\mathbf{p}$; the computer calculates via $\mathbf{H}(\mathbf{p})$, evolving into physical state $\mathbf{p}'$; the final state is represented as the final abstract model $m_{\mathbf{p}'} =_{\varepsilon} m'_{\mathbf{p}}$; this is decoded as the solution to the problem, A'. The instantiation, physical evolution, and representation together implement the desired abstract computation $C_{\mathcal{T}}(m_{\mathbf{p}})$. (From now on we omit the dashed line separating the physical and abstract world, and rely on the different shaped boxes to indicate what components lie in which domain.)

Ensuring that the diagram ε-commutes is a process of debugging the physical system, including how it is instantiated (engineered, programmed and provided with input data), and how its output is represented. This shows another key difference from the science cycle: there the diagram is made to ε-commute by instead debugging the abstract model.

The most important use of a computing system is when the abstract outcome $m'_{\mathbf{p}}$ is unknown: when computers are used to solve problems. Consider as an example the use of a computer to perform the abstract arithmetical calculation $2 + 3$. The desired final abstract state, $m'_{\mathbf{p}} = 5$, is not calculated directly. Instead, the user infers that final state by using the physical evolution of the computing device and the representation of its final physical state, which yields $m_{\mathbf{p}'}$; confidence in the technological capabilities of the computer and the correctness of the instantiation allows the user to exploit $m'_{\mathbf{p}} =_{\epsilon} m_{\mathbf{p}'}$ to infer that the observed result is the desired result.

This use of a physical computer is the compute cycle, Fig. 3: *the use of a physical system* (the computer) *to predict the outcome of an abstract evolution* (the computation).

2.5 Generality of AR theory

Nothing in the above definition requires the physical computer to be digital, or electronic, or universal, or pre-existing. The computer could be a continuous analogue device; it could be a mechanical or organic device; it could be a hard-wired device with limited capabilities; it could be a 'one-shot' device constructed for a particular computation. It simply needs to be sufficiently powerful, sufficiently accurate, and instantiatable, to perform the RE's desired computations: the relevant squares must exist, and must be known to ε-commute for the desired computations.

And, of most relevance here, nothing in the above definition requires the RE to be a human, or conscious, user. We now show how to model the RE in the same context as the computing system.

3 Including the RE in the model

3.1 Overview

As mentioned above, the representational entity (RE) supports the representation relation $\mathcal{R}$. Although it does not appear explicitly in the compute cycle of Fig. 3, it is the *physical entity* that 'owns' the abstract problem A and 'desires' the abstract solution A'.

To help clarify the issues, consider a (human) RE who has the problem "I have two apples in my left hand, and three in my right hand; how many apples do I have in total?" We model this physical RE's problem, how they encode it as a computational problem, how this is instantiated in a physical computer, how the computer finds the answer, how the answer is represented back as an abstract computational result, and how that result is decoded as an answer to the RE's problem.

In this section, we add the RE to the overall model of physical computing as defined in AR theory. As before, we have objects in two domains: the physical RE, and our abstract model of the RE. (We refer to 'our' model, to indicate that the abstract model of the RE is not a model that the RE has constructed about itself, but a model that we have constructed to explain the RE's representational and computational behaviours. There are several levels of indirection at play here, explained in more detail in Sect. 5.2.) First we show how we model the RE in a manner analogous to how we model a physical computation (Sect. 3.2). Then we show how to integrate the RE and full physical compute cycle models, and how to interpret various parts of the resulting model (Sect. 3.3).

3.2 The physical RE

The RE is a physical system $\mathbf{p}_{RE}$ (Fig. 4). The relevant part of the RE here is the physical states that it uses to represent its abstract problem A. Our abstract model of these relevant parts in this and following AR theory diagrams is $m_{\mathbf{p}_{RE}}$.

We model the computational system as before. There is our abstract model $m_{\mathbf{p}_c}$ that forms the 'specification' of the

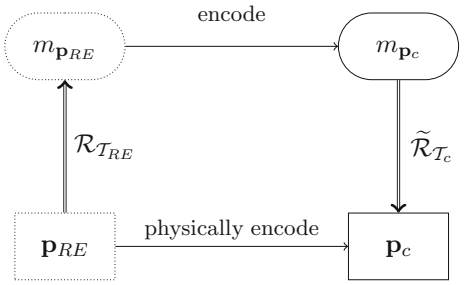

Fig. 4 The relationship between the physical representational entity $\mathbf{p}_{RE}$ and the physical computer $\mathbf{p}_C$ via abstract models of each. There is an *encoding* of the abstract model $m_{\mathbf{p}_{RE}}$ into $m_{\mathbf{p}_C}$. In a correctly working system, this encoding is appropriately implemented by the respective physical systems: the square should ε-commute. Note that the models of the RE and the computer are potentially with respect to different theories

RE's problem $m_{\mathbf{p}_{RE}}$ encoded as a computational problem. (This is the model $m_{\mathbf{p}}$ in Fig. 3.)

The computer's physical state may incorrectly implement the RE's encoded problem, in which case the physical state needs to be modified; the RE is using an engineering model to specify $\mathbf{p}_c$ (see Sect. 5.1 for the distinction between an engineering and a scientific model). However, our model $m_{\mathbf{p}_c}$ of the RE's encoded problem may incorrectly represent the RE's problem: we are using a scientific model to capture the existing RE and its computer. We model 'the RE's problem being physically encoded into the computer' (bottom line of Fig. 4) as '$m_{\mathbf{p}_{RE}}$ being *encoded* into the computational model $m_{\mathbf{p}_c}$' (top line). There is no guarantee that such an encoding is possible: not all problems are computable.

The two representation/instantiation relation arrows in Fig. 4 are with respect to two different theories. The representation $\mathcal{R}_{T_{RE}} : \mathbf{p}_{RE} \rightarrow m_{\mathbf{p}_{RE}}$ is based on the theory of how the physical RE forms abstract problem specifications; the instantiation $\widetilde{\mathcal{R}}_{T_c} : m_{\mathbf{p}_c} \rightarrow \mathbf{p}_c$ is based on the theory of how the physical computer implements abstract computations.

In a correctly implemented computer, the diagram in Fig. 4 should ε-commute: the instantiated state of the physical computer should correctly mirror the desired state of the physical RE: it should *physically encode* the desired state. The establishment of this physical encoding link is part of the engineering process of instantiating the physical computer.

During the execution, this physical encoding link is not necessarily established immediately. There may be some delay, for example in updating a record to reflect reality, or in opening or closing a valve to reflect changed demand. In, for example, a mechanical control system, with feedback, there can be an immediate coupling: the behaviour of the physical controlled system (a proxy for the RE, see Sect. 6) changes its state, which is directly communicated to the physical controller though their physical mechanical coupling. We do not consider this aspect further here, although it is a key feature of correctly-engineered computational 'mirror worlds' and of feedback control systems.

3.3 The physical RE in the compute cycle

We can now add this physical RE layer to the previous compute cycle. See Fig. 5 for the full compute cycle including the representational entity. Notice how the RE adds another dimension (cube instead of square) to the diagrams. Each dimension is a level of indirection or representation.

The full compute cycle involves traversal of many faces and edges of the displayed cube. Each face has its own place in the model.

Consider again a (human) RE who has the problem "I have two apples in my left hand, and three in my right hand; how many apples do I have in total?"

Back face; RE's view of the computation (Fig. 6): the RE's desired states, starting from a problem state (physical brain state with a problem, $\mathbf{p}_{RE}$; abstract initial state representing the brain-state's problem, $m_{\mathbf{p}_{RE}}$, "how many apples?") and resulting in a solution state (abstract final state, $m'_{\mathbf{p}_{RE}}$, "five apples!"; physical state, $\mathbf{p}'_{RE}$, a brain state that captures that abstract solution). There is no direct path from initial to final state, either abstractly or physically, as

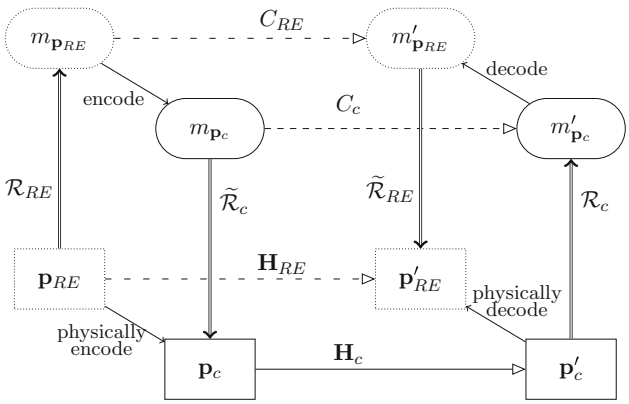

Fig. 5 The full compute cycle including the representational entity and the physical computer. The desired change in the RE's state, from posed problem to perceived solution, is $\mathbf{p}_{RE} \rightarrow \mathbf{p}'_{RE}$. The physical computer performs $\mathbf{p}_c \rightarrow \mathbf{p}'_c$. The full compute cycle from AR theory is: represent RE's physical state $\mathbf{p}_{RE}$ (desired computation) as abstract model $m_{\mathbf{p}_{RE}}$; encode to computational model $m_{\mathbf{p}_c}$; instantiate into physical computer state $\mathbf{p}_c$; physical computer evolves to final state $\mathbf{p}'_c$; represent physical solution as abstract computational solution $m'_{\mathbf{p}_c}$; decode to final abstract problem solution $m'_{\mathbf{p}_{RE}}$; which models the instantiation of the final state of the RE. Each set of squares (between representational entity and physical computer, and across the compute cycle) should ε-commute

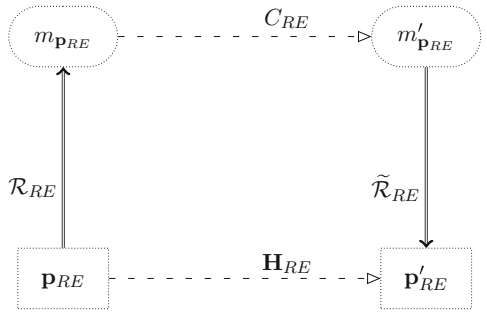

Fig. 6 The RE's view of the problem solution (back face of Fig. 5). The RE has an initial physical state $\mathbf{p}_{RE}$, modelled as $m_{\mathbf{p}_{RE}}$. It has a desired final state $\mathbf{p}'_{RE}$, modelled as $m'_{\mathbf{p}_{RE}}$. Both the horizontal arrows are dashed, as they are implemented in a different medium: the computer

a separate computer is used to achieve the desired state changes.

Left face; encoding the problem (Fig. 4): the RE's initial physical and abstract state encoded into the computer's initial physical and abstract states. The RE's abstract problem $m_{\mathbf{p}_{RE}}$ of "how many apples?" can be encoded as the computer's initial abstract state $m_{\mathbf{p}_c}$ "$2+3$". This is instantiated as the computer's initial physical state $\mathbf{p}_c$, $\boxed{2+3}$. The RE $\mathbf{p}_{RE}$ physically encodes the problem in the computer's initial state $\mathbf{p}_c$ by, for example, pressing the keys labelled $\boxed{2}$ then $\boxed{+}$ then $\boxed{3}$. (How this human RE manages to press the keys, given the apples they are currently holding, is an exercise left to the reader.)

Front face; compute cycle (Fig. 3, which also includes the back face RE abstract models as its 'abstract problem' components): the original simple AR theory compute cycle, ignoring the role of the RE. The abstract computational problem $m_{\mathbf{p}_c}$ is instantiated in the computer's initial physical state $\mathbf{p}_c$, $\boxed{2+3}$. Physical evolution is initiated by pressing $\boxed{=}$, and the physical computer evolves as given by its physical structure, $\mathbf{H}_c$, which results in the final physical state $\mathbf{p}'_c$ of $\boxed{5}$. This is represented as the final abstract state $m'_{\mathbf{p}_c}$ of "5". These three steps (instantiation, physical evolution, representation) implement the desired abstract computation C_c: "$2+3=5$".

Right face; decoding the solution (Fig. 7): the RE's final physical and abstract state decoded from the computer's final physical and abstract state. The computer's final physical state $\mathbf{p}'_c$ (some kind of pattern of lights in the shape of a figure 5) is represented as the final abstract state $m'_{\mathbf{p}_c}$ of "5". This is decoded to the RE's final abstract state $m'_{\mathbf{p}_{RE}}$ of "five apples!". The RE's final physical brain state $\mathbf{p}'_{RE}$ is an instantiation of this, physically achieved by the RE looking at and physically decoding the output from the computer.

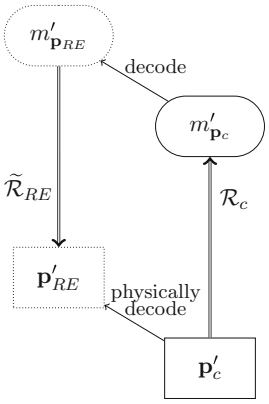

Fig. 7 Decoding the solution from the computer to the RE (right face of Fig. 5). The final state of the computer, $\mathbf{p}'_c$, is represented as the final abstract state $m'_{\mathbf{p}_c}$; this is decoded to the final abstract state of the RE, $m'_{\mathbf{p}_{RE}}$; and instantiated as the RE's final physical state. This is the model of the physical decoding lower arrow, achieved by the RE physically interrogating the computer

Top face; abstract use of a computer (Fig. 8): the purely abstract view of the (modelled) RE encoding its problem into a (modelled) computation, and decoding the desired solution. There is no direct path from initial to final abstract states as the physical computer is used to achieve the desired abstract state changes. In terms of classical refinement theory (Clark et al. 2005; He et al. 1986), C_{RE} can be thought of as the 'global-to-global' requirement (although here this need not be captured in a formal manner), with "encode, computation C_c, decode" corresponding to the "initialisation, operation, finalisation" steps.

Bottom face; physical use of a computer (Fig. 9): the purely physical view of the RE encoding its problem in a physical computer, and decoding the desired solution. That this is a *computation*, rather than some other activity, is established by the abstract models and the various ε-commuting squares.

The full compute cycle for the apples example is shown in Fig. 10. All of these relationships must be correctly implemented and modelled (the relevant squares containing encoding, decoding, instantiation, and representation must ε-commute) for the actual physical RE final state $\mathbf{p}'_{RE}$ to be the desired physical RE final state, that is, for the physical computer to have been used correctly, and for it to have performed correctly, to solve the RE's problem.

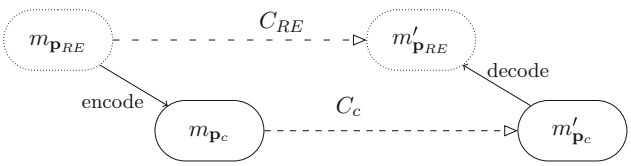

Fig. 8 The abstract model of the RE's use of the computer to solve its problem (top face of Fig. 5). The RE has an initial abstract state $m_{\mathbf{p}_{RE}}$; this is encoded into the initial abstract state of the computer $m_{\mathbf{p}_c}$. The computer performs its calculations to produce its final state $m'_{\mathbf{p}_c}$, which is decoded to produce the desired final state of the RE, $m'_{\mathbf{p}_{RE}}$. Both the horizontal arrows are dashed, as they are implemented in a different medium: the physical computer

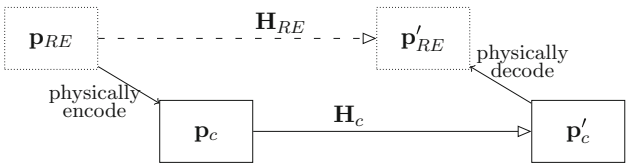

Fig. 9 The physical system of the RE's use of the computer to solve its problem (bottom face of Fig. 5). The physical RE has an initial physical state $\mathbf{p}_{RE}$; this is physically encoded into the initial physical state of the computer $\mathbf{p}_c$. The computer evolves over time to produce its final state $\mathbf{p}'_c$, which is decoded to produce the desired final state of the physical RE, $\mathbf{p}'_{RE}$

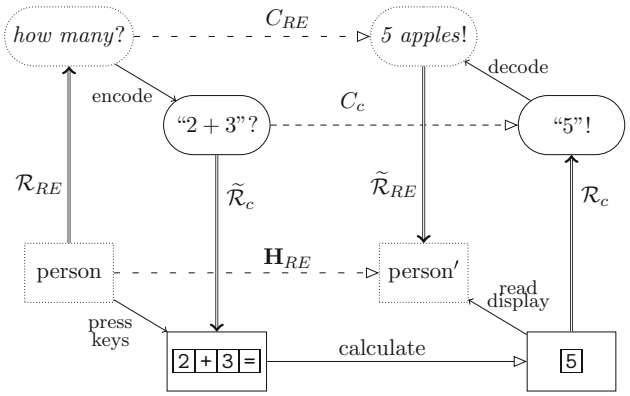

Fig. 10 The full compute cycle for the RE using a calculator to determine their apple total

4 Examples

4.1 Intrinsic computing: 'mental arithmetic'

The person with the apples does not need to use an external calculator; they can add up in their head, they can compute intrinsically. (This also solves their problem of pressing keys with their hands full of apples.) Figure 11 illustrates this. Here, the RE (back face) and the abstract model (top face) are as in Fig. 10, but the physical calculator is different. Instead of pressing keys on an external calculator, the RE 'sets' their own brain into the appropriate state, executes the calculation internally, and then 'reads' the resulting state. This uses an evolved, rather than engineered mechanism: the brain. It has been engineered (trained) to some degree through schooling, and there have been sufficient 'experiments' (exams) in the past to have confidence that the error ε is sufficiently small (zero) for simple calculations.

4.2 Intrinsic computing in bacteria

Humans are not the only organisms who can compute intrinsically. Figure 12 shows the RE and the compute cycle in the case of the problem of bacterial computing. This example was originally studied in Horsman et al. (2017a) to demonstrate that it is possible to have computation with a non-human RE. However, without the explicit modelling of the bacterial RE, it resulted in a somewhat circuitous description. With the RE here explicitly present, the model is much clearer.

The physical RE, $\mathbf{p}_{RE}$, is a bacterium, with a receptor at the front, and a flagellar motor at the back. In the absence of input at the receptor, the motor is off; input causes the motor to switch on, propelling the bacterium towards food. (As ever, the biology is more complicated than this; Horsman et al. (2017a) should be consulted for further biological details.) The abstract problem, C_{RE}, that the RE wants to solve is "if there is food, move towards it". This is encoded as the abstract computational problem C_c: "if there is a signal, switch the motor on". The abstract signal is instantiated as a particular chemical; the physical *RE* physically encodes the reception of exterior food as the presence of an internal chemical, chem X.

This chemical physically propagates through the bacterium, undergoing transformation via a biochemical pathway, such that another chemical becomes present at the rear. The presence of this other chemical, chem Y, is represented as switching on the (abstract) motor, which is decoded as the answer to the bacterium's problem: to

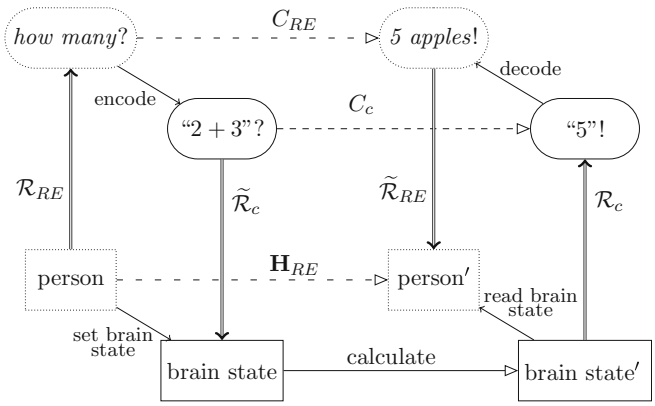

Fig. 11 The full compute cycle for the RE using their intrinsic calculational ability to determine their apple total

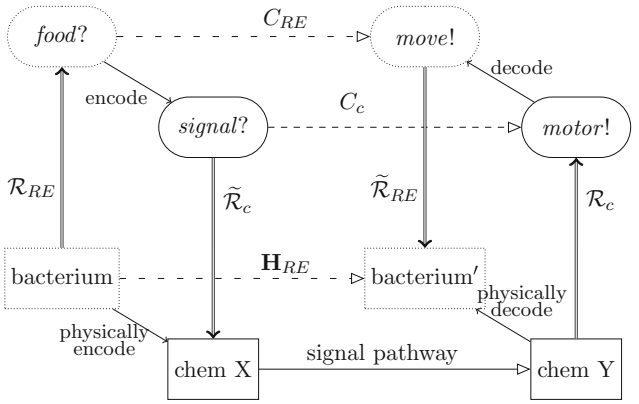

Fig. 12 The full compute cycle for the bacterial intrinsic computing system. See text for details

move. It is physically decoded as activating the flagellar motor.

The bottom face of this bacterial-compute cube shows the purely physical computing: The bacterial RE physically encodes the detection of food by its receptor as a chemical chem X; the biochemical pathway moves and transforms this chemical signal to the rear where it appears as chem Y. The resulting chemical is physically decoded: it attaches to and activates the flagellar motor. The physical problem, of detecting food and moving towards it, has been solved.

That this is indeed a *computation*, rather than a purely physical process, is argued in Horsman et al. (2017a): chem X, chem Y, and the pathway are in some sense 'arbitrary' (they comprise different molecules in different bacterial species), and so it is not their specific *physical* properties, but their *representational*, informational properties, that are being exploited. We are able to model the part of the bacterium that represents the problem as $m_{\mathbf{p}_{RE}}$, and the part that encodes into the computer $m_{\mathbf{p}_c}$ in a way that convincingly contains the right sorts of representation. With representation (and hence a representational entity) identified, we can conclude that there is abstract *data* being processed, not mere physical *material* being exploited. With ε-commuting diagrams present, we can conclude that computation is present.

There are similarities with the argument here and the field of biosemiotics, which talks of representations and signs, and postulates that "The semiosic–non-semiosic distinction is coextensive with the life–nonlife distinction" (Kull et al. 2009). For the presence of computation, we require representation, that the chemical be a 'signal' or 'sign' beyond its mere physical properties; we determine that it is so by showing a different chemical can provide the same signal.

Horsman et al. (2017a) also discuss the energy transport process in photosynthesis, as an example of a process that is *not* computation under our definition. There is no abstract information processing occurring, no intrinsic representational activity, no signal, simply physical energy transport.

4.3 Extrinsic computing with bacteria

Not all computing done by bacteria is intrinsic: other REs can exploit bacterial functions to perform extrinsic computation for their own ends. Various gene-engineered bacteria and cells have been proposed as computational devices, see, for example, Amos et al. (2015); Gardner et al. (2000). Consider a simple bacterium gene-engineered to perform some computational task, and its used by a human RE, Fig. 13.

The person has a question that they can encode into an input to their bacterial computer; it computes to produce a final state that is represented by an output; that output is decoded into the answer, and the person's brain state updates to include the answer to their question.

That this is a computation that involves representation can be determined by interrogating the user as RE. So, unlike the intrinsic bacterial computation of the previous section, the bacterium is the computer and not the RE, and there is no need to demonstrate the existence of representation independently.

5 Modelling issues

There are multiple models used in our description of computation, which have some subtle differences, described here.

5.1 Scientific *v* engineering models

There are two kinds of model at play in these diagrams and our analysis.

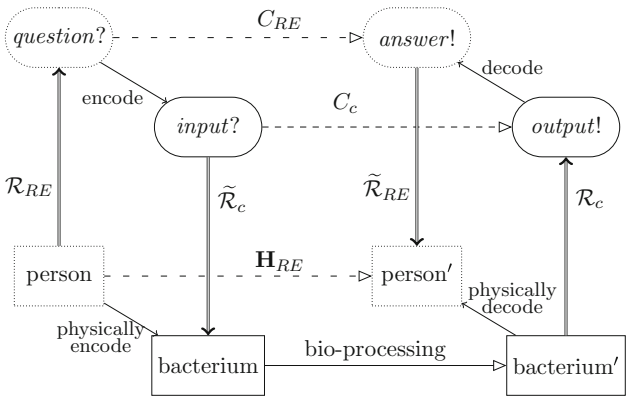

Fig. 13 The full compute cycle for the bacterial extrinsic computing system. See text for details

Firstly, there are scientific models, descriptive representations of physical reality. In the diagrams, the model and physical system are linked with an upward pointing representation arrow. If there is a discrepancy between such a model and reality, the model needs to be adjusted to be a better fit.

Secondly, there are engineering models, prescriptive specifications to be instantiated in physical reality. In the diagrams, the model and physical system are linked with a downward pointing instantiation arrow. If there is a discrepancy between such model and reality, reality needs to be re-engineered to be a better fit.

The presence of engineering models in the compute-cycle gives us another criterion for deciding on the presence of computation, rather than mere physical action. If ε is too large, if the diagram does not sufficiently commute, what is changed, the model or the instantiation? In an engineered (or evolved) computational system, it will be the instantiation.

- Using a calculator: if the user enters $\boxed{2+3=}$ and the display says $\boxed{11}$, the calculator has to be fixed (or maybe the user has to realise the display is in base 4).
- Mental arithmetic: if the person gets an answer other than "5", they need to go on a remedial arithmetic course.
- Intrinsic bacterial computing: if chem Y fails to bind to the flagellar motor, the bacterial line needs to evolve another chemical.
- Extrinsic bacterial computing: if the final bacterial state is not the desired one, the bacterium needs to be re-engineered.

There are additional places where errors may occur: the RE may conceptualise their problem incorrectly, may instantiate it in the computer incorrectly, or may decode the result incorrectly. All of these require model or instantiation changes, too.

5.2 Whose model is it anyway?

The models in the AR diagrams are made by us as AR theorists external to the computational system. These are our attempt to capture certain models implicitly or explicitly used by the REs in order for them to exploit the computational system.

In Fig. 8 the four models are *our* (AR theorist) models of the physical layer components of Fig. 9. The RE may use models in order to compute, but does not in general itself construct AR diagrams.

So the model $m_{\mathbf{p}_{RE}}$ is *our* model of the RE's problem, constructed to help us determine the presence and nature of the computation. The model $m_{\mathbf{p}_c}$ is *our* model of the RE's model of the computer and encoding; the RE's own model is also part of its physical state $\mathbf{p}_{RE}$. (The RE's model of $m_{\mathbf{p}_c}$ is an engineering model, as indicated by the direction of the arrow; our model of the RE's model is a scientific model.) Similarly for the other models in Fig. 8.

In the case of a human RE, we may be able to interrogate them about their own models to help us build our models; for non-human users, we need to take a more indirect route. We elide this complexity in the figures, for clarity. However, we must recognise the possibility that our models may incorrectly capture the RE's physical state and problem, and its model of the computer (we have failed to correctly model the RE/computer system), in addition to the possibility that the RE may have incorrect models (the RE is mistaken about its understanding of the computer). When a discrepancy is discovered in our formulation of the situation, we need to modify our descriptive models of the RE's descriptive and prescriptive models.

6 Conclusion

We have shown how the RE in AR theory can be incorporated into the compute cycle, and how this can illuminate the physical RE using a physical device as a physical computer. The RE does not need to be a human brain: an example here shows intrinsic computing by a bacterial RE; the case does have to be carefully argued, by demonstrating the arbitrariness of the representation. This example demonstrates how computing, whether conventional or unconventional, can be broader than human use of computers (external or brain-based), but is narrower than pan-computationalism, in requiring the existence of an RE in addition to the computer itself.

It may be that the RE does not even need to be organic, or 'alive'; it might potentially appear in the loop as an engineered 'proxy' for the ultimate RE, for example, the plant (proxy RE) in a control system using the controller (physical computer) to maintain itself in a particular behaviour. In future work we will investigate how far the concept of the RE can be removed from a living user.

Acknowledgements We thank our colleagues Dom Horsman, Tim Clarke, and Peter Young, for illuminating discussions. VK was funded by UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/L022303/1.

References

Amos M, Axmann IM, Blüthgen N, de la Cruz F, Jaramillo A, Rodriguez-Paton A, Simmel F (2015) Bacterial computing with engineered populations. Philos Trans R Soc A Math Phys Eng Sci 373(2046):20140218

Clark JA, Stepney S, Chivers H (2005) Breaking the model: finalisation and a taxonomy of security attacks. In: Refine 2005 workshop, Guildford, UK, April 2005. Vol. 137(2) of Entcs, 225–242. Elsevier

Gardner TS, Cantor CR, Collins JJ (2000) Construction of a genetic toggle switch in Escherichia coli. Nature 403:339–342

He J, Hoare CAR, Sanders JW (1986) Data refinement refined (resumé). In: Proceedings of European symposium on programming, ESOP 86. Vol. 213 of LNCS, pp 187–196. Springer

Horsman DC (2015) Abstraction/representation theory for heterotic physical computing. Philos Trans R Soc A 373:20140224

Horsman C, Stepney S, Wagner RC, Kendon V (2014) When does a physical system compute? Proc R Soc A 470(2169):20140182

Horsman D, Kendon V, Stepney S, Young P (2017a) Representation and reality: humans, animals, and machines. In: Dodig-Crnkovic G, Giovagnoli R (eds) Abstraction and representation in living organisms: when does a biological system compute?, vol 28. Springer, Berlin, pp 91–116

Horsman D, Stepney S, Kendon V (2017b) The natural science of computation. Comms ACM 60(8):31–34

Horsman D, Kendon V, Stepney S (2018) Abstraction/Representation Theory and the Natural Science of Computation. In: Cuffaro ME, Fletcher SC (eds) Physical perspectives on computation, computational perspectives on physics. Cambridge University Press, Cambridge, pp 127–149

Kull K, Deacon T, Emmeche C, Hoffmeyer J, Stjernfelt F (2009) Theses on Biosemiotics: Prolegomena to a Theoretical biology. Biol Theory 4(2):167–173

Stepney S, Kendon V (2019) The role of the representational entity in physical computing. In UCNC 2019, Tokyo, Japan, June 2019. Vol. 11493 of LNCS, 219–231. Springer