

Computation, individuation, and the representation condition

Mark Sprevak

King's College, Cambridge CB2 1ST, UK
Tel: +44 (0)7752 346171, Fax: +44 (0)1223 334554
E-mail: mds26@cam.ac.uk

29th June 2008

1 Introduction

What makes a physical process a computation? What is the difference between a computation and any other process? Under what conditions are two computations the same or different? These are among the key questions that a philosophical theory of computation should answer. The detailed shape of the answers, and that of the corresponding account of computation, is not yet clear. Yet there seem to be certain features that any reasonable account of computation should possess. What has been labelled 'the received view' is that computation must involve representation.¹ According to this view, a necessary condition on any process counting as a computation is that it possess representational content. The received view has come under two influential attacks, from Egan (1991, 1992, 1994, 1995) and Piccinini (2008). Egan argues that one should understand computation in formal mathematical terms, Piccinini that one should understand computation functionally.² In this paper, I focus on Egan's argument. I defend the received view by arguing that Egan's argument against it fails. The focus is on Egan, but there are points of contact throughout with Piccinini's argument and these will be noted in passing. A full discussion of Piccinini's argument has to wait until another occasion.

¹Philosophers with views as divergent as Churchland (1986); Crane (2003); Cummins (1989); Dennett (1971); Fodor (1998); Pylyshyn (1984); Searle (1992) hold that computation involves representational content.

²His theory of computation is based on recent accounts of mechanism developed by Bechtel and Richardson (1993); Craver (2007); Glennan (2002); Machamer et al. (2000).

The purpose of this paper may appear overly negative: to show that Egan's attack against the representation condition fails. However, the argument is constructive in a number of important areas. Demonstrating why Egan's argument fails requires defending a number of significant positive claims about computation. First, that a distinction should be made between the concept of computation employed by mathematical computation theory and that used in describing the performance of a computation by a physical system. Second, even if one wishes, as Egan and Piccinini do, to take mathematical computation theory as a model for all computation talk, appeal to representational content is inescapable when attributing computations to physical systems. This produces an internal critique of Egan's position: even by her lights she is committed to the representation condition. Third, *contra* Egan and Fodor, there is no conceptual link between computational psychology and individualism. There is no reason why a computational psychology should be individualistic, or if it were to involve representation, why it should only involve narrow content. Methodological solipsism is no part of the computational theory of mind. Fourth, in addition to the internal critique of Egan, some additional arguments are given for why computation should involve representational content.

In Section 2, I present three arguments against the representation condition and show why they fail. The first argument is not endorsed by Egan but it has widespread currency in the philosophical literature, and is worth considering if only to get it out of the way. Egan's arguments receive a more detailed treatment. Egan's first argument is based on the interpretation of Marr's theory of vision. Egan argues that Marr's theory of vision—a paradigm of computational explanation—does not posit representational content, and hence that not all computations need involve representational content. Egan's second argument involves a dilemma concerning narrow content. She claims that anyone who accepts computational psychology must accept either an unpleasant commitment to narrow content, or drop the representation condition entirely. I argue that Egan's first argument fails, and that her dilemma can be resisted. In Section 3, I briefly turn to the positive argument for the representation condition. This section is not intended to be a full-fledged defence of the representation condition, that would require a paper in itself, but it does highlight what I take to be the key intuitions that should underlie such a defence. The overall purpose of the paper is to show that the representation condition is alive and kicking. As a received view, it may appear apt for debunking, but in this case, the received view is simply true.

Before proceeding a number of qualifications should be mentioned.

First, the disputed claim about the relationship between computation and representation is the claim is that representation is essential to computation,

namely,

(R) Computation essentially involves representational content

Egan and Piccinini allow computations to involve representational content, but they argue that such features are accidental to a system's computational nature, and have no bearing on its computational identity. My claim is that representational content is a necessary feature of computation that does crucial work in determining the facts of computational identity. On this view, representation is still only part of the story about computation: there are other conditions that a computation should satisfy, and there are more conditions on computational individuation than just those involving representational content. However, representational content does much of the hard work in answering the questions above that motivate an account of computation. Consequently, it is a feature of computation that should be of special interest.

Second, the dispute over whether computation essentially involves representational content is often phrased in terms of a consequence that such a view might have: that the computations involved in cognition essentially have their intentional content.³ I wish to avoid phrasing the debate in terms of essential intentional content. The question of whether computation is committed to intentional content introduces a number of requirements that appear to go beyond (R). One would not wish to foreclose these issues when considering the status of (R). First, intentional contents arguably require the essential involvement of cognitive agents, but one would not wish to foreclose the question of whether computations can take place without involvement of cognitive agents. Second, intentional contents have a mode of presentation, an aspectual shape, as well as an object. But one would not wish to foreclose the question of whether the representations involved in computation must have a mode of presentation as well as an object or referent. Third, intentional states play complex and rich functional roles in our folk psychology. But one would not wish to foreclose the question of whether the representations involved in computation must be apt for playing the same roles in our psychology (e.g. whether they have propositional structure). These are all important questions, but they are posterior to the question of whether computation involves any kind of representational content at all. In what follows, I will attempt to avoid foreclosing these issues by phrasing the dispute with the more neutral term 'representational content'. This is to be understood as a place-holder that is satisfied by any kind of representational content. Roughly, a *representation* should support a basic notion of aboutness or reference. A representation should link an entity and its content,

³Burge (1986); Egan (1991, 1992, 1994, 1995); Fodor (1980); Segal (1989); Shagrir (2001).

such that the entity represents its content. It is not required to support much more. In particular, it is not presupposed that any additional requirements associated with intentional states are satisfied.

Third, some of the literature phrase the dispute in terms of the nature and individuation of computational *states*. As will be argued in Section 2.3, questions about individuation of computations should be phrased in terms of *processes*: the basic units of computation are computational processes, and individuation of computational states is parasitic on individuation of computational processes. This is more than just a matter of convention or terminology. As argued in Section 2.3, excessive focus on computational states has led to the unjustified assumption that computations essentially involving broad content is objectionable. In what follows, the terms ‘computational process’, ‘computation’, and ‘computational system’ will be used interchangeably. Unless noted otherwise, what is meant by these terms is the implementation of a computation by a physical system.⁴

2 Arguments against the representation condition

This section presents three arguments against computation involving representation. The first argument has few defenders, but it is nevertheless worth considering since its thinking has widespread currency in the philosophical community.⁵ The other two arguments are carefully developed by Egan and warrant more attention.

2.1 Distinction between dynamics and individuation

A seductive, and commonly held, line of thought about computation is as follows. Computations are formal, their transitions are governed only by the syntactic character or ‘shape’ of the computational tokens. A computational token’s shape typically covaries with its semantic properties. But it is the shape that does the causal work in the transitions, not the semantic properties. Indeed, computation is a way in which a process can appear to be semantically-sensitive without spookily depending on what its tokens refer to. Therefore, semantic properties do no essential causal work in the transitions of computational processes. The conclusion is that semantic properties may ride along

⁴The restriction to physical systems is not essential. If non-physical systems perform computations, then similar concerns apply.

⁵One example is Searle (1990)’s ‘axiomatic argument’ against Strong AI: since computational operations are purely formal, computations need not have semantic content. Another is Stich (1983)’s argument that the computational theory of mind supports eliminativism about intentional content because, if mental processes are computations, their representational content is explanatorily irrelevant to their dynamics (p. 193).

with computations, but they are not among their essential properties. Therefore, representational content is not essential to computation.

Two problems should be noted with this argument.

First, the argument mistakes the form of dependence on representational content at issue in the debate over the representation condition. An advocate of (R) typically does not claim that the *causal dynamics* of computations depend on representational content. Her claim is that the *individuation* of computations depends on representational content. It is facts about the individuation and identity of computations, not about their causal transitions, that are the battleground for the representation condition. If one cannot make sense of computational identity, or the computational/non-computational contrast without reference to representational features of computations, then representation is an essential feature of computation, regardless of its claimed irrelevance to causal dynamics. Therefore, pointing to the non-semantic nature of computational transitions simply fails to engage with the main content of the claim that computation essentially involves representational content.

A second problem is that the above argument relies on a questionable notion of 'essential causal work'. It is not obviously correct that if one property (or cluster of properties) 'does the causal work' of another, then that latter property is causally irrelevant. As debates over Kim's exclusion argument illustrate, to assume such a principle is universally valid is to commit oneself to a strong form of reductive materialism where the only causally relevant properties lie at the bottom level of physics, if such a level exists. Unless one is sanguine about this possibility, the argument above that semantic properties of computations are causally irrelevant *because* their work is done by syntactic properties should be regarded with suspicion. Some further justification would be required to show that semantic properties are really irrelevant to the causal dynamics.

Furthermore, there appear to be positive reasons for thinking that there are connections between facts about individuation and causal dynamics. Facts about causal dynamics include, *inter alia*, facts about which events (relata) are involved in the causal dynamics. The events that exist, and whether those events include the tokening of computational states, depend on facts about the individuation of computations. If one wishes to allow computational states to have causal powers *qua* computational states, then appeal to their individuating properties is required to account for the causal dynamics of the system. If those individuating properties include their semantic properties, then their semantic properties will essentially figure in the causal dynamics. Therefore, although computational transitions may not require any spooky dependence on semantic properties, that does not mean that semantic properties are causally irrelevant in the overall causal dynamics of the system. If computational states

qua computational states are not to be causally inert, then facts about their computational individuation cannot be cleanly separated from the facts about their causal dynamics.

2.2 Egan's argument from interpretation of Marr

A significant part of the debate over whether computation involves representational content has centred around the correct interpretation of Marr's theory of vision. Egan's first argument against the representation condition relies on the interpretation of Marr. Piccinini (2008) raises worries about this methodology: Marr's theory is just one among many, it may turn out to be wrong, and there may be no fact of the matter about how Marr intended his theories to be interpreted.⁶ Egan argues that Marr's theory gives us reason to think that computation does not involve representational content. It is worth noting that Piccinini's objections fall short of showing that interpretation of Marr's theory lacks probative force in this context. First, Marr's theory is a paradigm of a successful computational theory in psychology. It provides a model of what a theory should look like, even if it is false. If such a paradigm can be interpreted without appeal to representational content, then that is evidence one need not be committed to computation involving representational content. Second, many of the questions raised for Marr's theory can be recast for other theories in computational psychology, so it is unclear whether a worry about narrow scope has bite. Third, even if there is no fact of the matter about how Marr would have interpreted those aspects of his theories concerning the relationship between computation and content, a consistent interpretation should still be possible. Piccinini's worry appears to be that Marr's opinions on this point were not fully formed, not that Marr's underlying computational theory is hopelessly confused or inconsistent. If the most plausible interpretation of the computational theory does not make essential use of representational content, then that is evidence that computation need not involve representational content, regardless of Marr's own interpretation.

A number of participants in the debate defend the view that Marr's theory of vision makes essential use of representational content.⁷ Egan (1995) argues that this is wrong. She claims that although a *complete* explanation of visual processes will typically advert to representational content, the *computational core* of such a theory is purely formal. Just as an explanation of the ability of a sand shark to detect its prey using electric fields adverts to both

⁶Piccinini (2008), pp. 207–208.

⁷Their primary concern is whether the representational content is wide or narrow. See Burge (1986); Davies (1991); Kitcher (1988); Shapiro (1997) for broad content, and Cummins (1989); Morton (1993); Segal (1989, 1991) for narrow content.

electromagnetic theory and the fact that in the shark's environment, animals, but not rocks, produce significant electric fields, without the latter fact being part of *electromagnetic theory*, so an explanation of vision will typically avert to representational features beyond its purely formal computational properties, without those representational features being part of the computational theory of vision.

What reason does Egan give for thinking that computational theories are purely formal? She starts by distinguishing Marr's three levels of computational description: (i) the *computational* level, which characterises the function computed by the system; (ii) the *algorithmic* level, which specifies an algorithm for computing that function; and (iii) the *implementation* level, which describes how the process is physically realised. Marr's computational level is sometimes assumed to be intentional and occasionally straightforwardly equated with Pylyshyn (1984)'s semantic level, Newell (1982)'s knowledge level, or Dennett (1987)'s intentional level. Egan claims that this is a mistake. To provide a description at Marr's computational level is not to adopt any kind of intentional or semantic stance, but to adopt the point of view of mathematical computation theory for describing the system. In the context of Marr's theory, the computational level should be understood in a purely *mathematical function-theoretic* way. Computational description is a characterisation of the functions computed by the various parts of the cognitive system in mathematical terms, not in terms of what those parts represent. For example:

I have argued that from a computational point of view [the retina] signals $\nabla^2 G * I$ (the X channels) and its time derivative $\partial/\partial t(\nabla^2 G * I)$ (the y channels). From a computational point of view, this is a precise characterization of what the retina does. Of course, it does a lot more—it transduces the light, allows for a huge dynamic range, has a fovea with interesting characteristics, can be moved around, and so forth. What you accept as a reasonable description of what the retina does depends on your point of view. I personally accept $\nabla^2 G$ as an adequate description, although I take an unashamedly information-processing point of view. (Marr, 1982, p. 337)

At the computational level, the retina should be described merely as computing the function $\nabla^2 G$. $\nabla^2 G$ is a mathematical function that maps two-dimensional arrays $I(x, y)$ to isotropic rates of rates of change of $I(x, y)$ at points (x, y) via convolution. $\nabla^2 G$ is a mathematical relation. A computational description of the visual system is a purely formal mathematical description of the mathematical function computed by the system. It is not a description of the visual system in terms of, for example, representations of light intensity

values and representations of shapes. It is neutral about what the inputs and outputs to the system represent or fail to represent, or indeed whether they have any representational content at all.

Egan claims that if the computational level need not involve representational content, then the algorithmic level need not involve representational content either. Therefore, the computational core of Marr's theory makes no essential use of representational content; it is a purely mathematical characterisation. In the above quotation, Marr says that the visual system may have other properties—and to his list Egan would presumably add representational properties—but those properties are not essential to the computation that the system performs. Hence, a paradigmatic case of a computation performed by a physical system—the computation that Marr attributes to the retina—does not essentially involve representational content. Therefore, (R) is false.⁸

What can be said in response to this argument?

First, a distinction should be drawn between mathematical computation theory and Marr's computational level. Egan interprets Marr's computational descriptions as of the same kind as the function-theoretic descriptions given in mathematical computation theory. Mathematical computation theory is a branch of pure mathematics. It describes relations between mathematical structures and objects. The 'computers' it considers are mathematical entities, not physical systems. From the perspective of mathematical computation theory, a Turing machine is not a physical system: it does not 'perform' a computation in the same sense as a physical system does. A Turing machine is typically identified with a set of set of mathematical symbols, e.g. with the quintuple $M = (Q, \Sigma, \Gamma, \delta, q_0)$, where Q is a set of state symbols, Γ is a set of numerals that can be used on the tape, B a special symbol that represents a blank, Σ a subset of $\Gamma - \{B\}$ called the input alphabet, δ is a partial function from $Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R\}$ called the transition table, $q_0 \in Q$ a special state called the start state, and the symbols L, R the direction of movement along the tape. The tape is another

⁸Like Egan, Piccinini (2008) takes mathematical computation theory as his model for how physical computation talk should be understood:

In [mathematical] computability theory, symbols are typically marks on paper individuated by their geometrical shape (as opposed to their semantic properties). Symbols and strings of symbols may or may not be assigned an interpretation . . . In these computational descriptions, the identity of the computing mechanism does not hinge on how the strings are interpreted . . . More generally, the whole mathematical theory of computation can be formulated without assigning any interpretation to the strings of symbols being computed . . . [These] considerations apply straightforwardly to ordinary, non-universal Turing machines, and any other computing mechanism whose behaviour is not *controlled* by a program. (Piccinini, 2008, pp. 211–212)

To the extent that Piccinini takes features of mathematical computation theory (including its non-semantic nature) to transfer straightforwardly over to physical computing mechanisms, his argument appears vulnerable to the objections raised below. Even if mathematical computation theory can avoid making essential use of representational content, physical computations cannot.

mathematical structure in which symbols can be kept in linear order.⁹ One can make sense of a Turing machine having ‘inputs’: the initial symbols on the tape. One can make sense of a Turing machine having ‘outputs’: the final symbols on the tape. But in each case the inputs and outputs are mathematical objects; typically, strings of numerals like 0 and 1. It should be emphasised that these inputs and outputs are not empirical *ink-marks*. They are mathematical entities, numerals understood as abstract objects. Typically, one does not care what these abstract entities turn out to be, so long as one can make sense of them being numerically distinct. The computations studied in mathematical computation theory are independent of how things are in the empirical world. They are independent of empirical ink-marks, just as they do not ‘take place’ in time (or depend on the nomological possibility of infinitely long tapes). ‘Computers’ in mathematical computation theory are mathematical entities that bear certain relations, studied by mathematical computation theory, to other mathematical entities, the functions they compute.

Questions about triangles in pure Euclidean geometry (‘Do the angles of a triangle add up to 180 degrees?’) are typically distinguished from questions of real-world geometry (‘Do the angles of a terrestrial triangle add up to 180 degrees?’). The first question is a matter of mathematical truth and settled by proof; the second is a matter of empirical facts and settled by measurement. Pure Euclidean geometry, by itself, says nothing about terrestrial triangles; the points, lines, and planes it studies are abstract mathematical entities. Similarly, mathematical computation theory, by itself, says nothing about physical systems: the ‘computers’ it studies are mathematical entities. Thinking about physical systems, like paper-tape machines, has a heuristic and propaedeutic value when learning mathematical computation theory and when thinking through mathematical computation theory problems; just as thinking about terrestrial triangles, lines, and planes also has a heuristic and propaedeutic value to those learning and studying Euclidean geometry. More importantly, a significant part of the motivation for studying these areas of pure mathematics, and the reason why they are classified as distinctively ‘geometrical’ or ‘computational’ is their potential applications to empirical systems. However, it should be clear that terrestrial entities are *not* the subject matter of the relevant mathematical claims. Mathematical computation theory does not say anything about empirical physical systems.

Therefore, mathematical computation theory does not, by itself, have the resources to explain how the visual system works. Consequently, Marr’s computational level cannot be straightforwardly identified with the function-theoretic descriptions given in mathematical computation theory. Marr requires some

⁹See Sudkamp (1998), pp. 259–260 for more on the definition of a Turing machine.

way of connecting his abstract mathematical descriptions to the nuts and bolts of physical reality.¹⁰ So how do mathematical objects, like $I(x, y)$ and (x, y) , get connected to the human visual system? In order to answer this question, Egan introduces the notion of a *realization function*:

... a realization function f_R ... maps equivalence classes of physical features of a system to what we might call “symbolic” features. Formal [computational] operations are just those physical operations that are differentially sensitive to the aspects of symbolic expressions that under the realization function f_R are specified as formal features. The mapping f_R allows a causal sequence of physical state transitions to be interpreted as a *computation*. (Egan, 1992, p. 446)

A realization function maps the nuts and bolts of physical reality to the mathematical objects in computation theory. It is the link between the abstract world of mathematical computation theory and the empirical world of the human visual system. For Egan, a realization function does the bulk of the work of computational individuation: it determines whether a physical system performs a computation, and if so, which computation it performs.¹¹ If a realization function obtains between a physical system and the formal entities involved in a mathematical computation, such that transitions in the physical system match those of the symbolic entities in the mathematical computation, then *that physical system performs that computation*. The realization function is, according to Egan, to be understood non-semantically: it associates physical states with mathematical entities independently of any representational content that those physical states might have. A realization function is no more than a mapping (a pairing) between classes of physical features of a system and abstract mathematical entities.

What can be said for this?

First, there is a problem with universal realization. If, as Egan proposes, the existence of an appropriate realization function is sufficient for a physical system to perform a computation, then almost every physical system performs every computation. A realization function, as described by Egan, is an isomorphism between a mathematical computation and a physical system, and the appropriate isomorphisms are cheap. Consider a desk; there are billions of particles inside a desk, and those particles undergo complex patterns of vibrational activity, thermal activity, electromagnetic activity, and so on, over time.

¹⁰Note that this is different from the question that Egan (1995) considers on pp. 189–194: how a formal computational account of the visual system can connect to explaining the *intentional* capacities of the subject. The question here is how a formal computational description can even *be about* the human visual system.

¹¹Egan (1994), p. 261; Egan (1992), p. 448.

There are billions of patterns of physical activity inside a desk. There is so much activity inside the desk that there is certain to be at least one pattern with a structure isomorphic to the structure of any finite mathematical computation one likes. Putnam (1988) shows that one does not even need a complex system like a desk to have universal realization on this type of view of computation. By just considering the phase space of a single particle as it evolves over time, one can construct a realization function that makes that system perform any finite computation one likes. Chalmers (1996) shows that the transitions involved can also support the relevant counterfactuals. There are qualifications to be made to these arguments, but the general problem presses against Egan. If the mere *existence* of a realization function is sufficient for a physical system to perform a computation, then almost every physical system performs every computation. A realization function pairs almost any physical system with any finite computation one likes.¹²

Clearly, something has gone wrong. If realization functions determine computational identity, then the computational identity of physical systems would be a trivial matter. In order to avoid triviality, some extra constraint on computational identity is required. It seems plausible to suggest that not every realization function determines a computational identity: some realization functions establish that their physical systems perform a computation, while others fail to secure computational status for their physical systems. This raises the question: What makes certain realization functions special? Why does the existence of some, but not other, realization functions suffice to establish a computational identity?

Two tempting answers to this question should be avoided.¹³ First, it cannot be that some realization functions fail to establish a computational identity because they are somehow ‘artificial’ or ‘too disjunctive’. For there are plenty of such realization functions that do establish computations. Think of the kinds of realization functions involved in electronic PCs: what is natural or direct about mapping the voltage levels 5 ± 0.4 V in a 01100001 pattern in certain capacitors in a machine to the symbol ‘a’? Such a mapping is neither obvious, nor carving the world at ‘natural’ joints. Yet it is the realization function employed by electronic PCs. Unless one denies that those systems perform computations, it is hard to object to a realization function being ‘artificially cooked up’ or ‘disjunctive’. Second, the reason why some realization functions

¹²Putnam (1988)’s argument is made in terms of finite state automata but it can be generalised to other computational formalisms. Chalmers (1996)’s argument requires that a physical system possess a ‘clock’ and a ‘dial’, but these conditions are easily satisfied.

¹³Variations on these answers are explored and endorsed in Cummins (1989), Ch. 8 in the context of ruling out unintended interpretation functions. See also Egan (1992), pp. 450–451; Egan (1995), pp. 192–193.

do not establish computational identity cannot be that it requires epistemic work on the part of the agent to construct that realization function.¹⁴ First, as noted above the realization functions of PCs often require considerable work on the part of the agent to construct, and sometimes the agent (the hardware designer) has to perform the computation herself in order to construct that function. Second, such a move would make the computation a system performs entirely a function of the epistemic powers and interests of the agents that investigate the system. There would be no other non-trivial constraint on the computation that a system performs. This would be to endorse a form of anti-realism about computation: facts about computation would be constrained, not by the systems investigated (which by themselves trivially perform every computation), but by facts about epistemic agents investigating those systems. Questions about how a realization function is *constructed* should not matter in this debate unless one wishes to endorse anti-realism about computation. What matters is why the *existence* of certain realization functions (no matter how we discover or construct them) is sufficient to establish certain computational identities. In brief, there is little hope of solving the problem by appeal to the *structure* or *epistemic origin* of realization functions.

The motivation for positing realization functions was to explain how Marr's talk of mathematical entities could connect to the empirical reality of the visual system. One might argue that Marr does not claim that the visual system performs a particular computation *simpliciter*, but that it performs a particular computation *under a particular realization function* F_R . This would avoid the problem above of Marr's claim being made trivially true. However, it would go beyond mere appeal to the truth of an isomorphism to explain how Marr's theory relates to the visual system. It would be to say that not only does that isomorphism obtain, but that it is also somehow special, or particularly relevant, to explaining the visual system. In what does the specialness of that particular realization function lie? Why is the computation that the retina performs $\nabla^2 G$, and not something else (as it would be under the unlimited number of other realization functions that are satisfied by the retina)? This is not something that Egan's account has the resources to answer.

However, a natural answer lies at hand. What makes a particular relation between the nuts and bolts of empirical reality and mathematical entities special is that those nuts and bolts *represent* those entities. Certain realization functions are special because they truly describe representation relations in the world. The retina performs the computation that Marr suggests because its inputs and outputs *represent* the relevant mathematical structures. Representation is the

¹⁴Cummins (1989) appears to endorse this answer as part of his condition that an interpretation function be 'direct' (pp. 102–105).

crucial link between the mathematical and empirical world. It draws the line between relevant and irrelevant realization functions.

Egan distinguishes between an interpretation function, f_I , and a realization function, f_R . According to Egan, an interpretation function is not essential to a computation: it assigns intentional content to symbolic states. A realization function is essential: it assigns mathematical symbolic entities to physical states. Egan (1994) criticises Morton (1993) for collapsing the two mappings f_I and f_R into one, but Morton in a sense is right: both mappings are representational. In one case, the representational contents are intentional contents, in the other, they are mathematical entities. Both face the problem of the relevant isomorphisms being cheap. Egan skirts around this issue in the case of interpretation functions, since on her view they are not essential to computational identity, and she argues that they are in any case fixed by independent facts about representation.¹⁵ However, she faces the issue full-force for realization functions. The same remedy as used for interpretation functions is required. The only way to winnow down the infinity of realization functions in a way that respects the flexibility and arbitrariness of computation is to make essential appeal to the notion of representation.

2.3 Egan's dilemma concerning narrow content

Egan's second argument against (R) involves an ingenious dilemma concerning narrow content.¹⁶ Egan argues that the nature of computation forces advocates of computational psychology into a dilemma. Either they can hold onto the representation condition but be committed to all cognitive computations involving narrow content (as Cummins (1989); Fodor (1980); Segal (1989) do), or they could give up the representation condition entirely. Egan argues that we should choose the latter option for two reasons. First, it is notoriously difficult to construct a plausible notion of narrow content. Second, it is hard to see how such a notion could both live up to its billing as a form of representational content and play a significant role in psychology, where content is typically specified in environment-specific, broad, terms. Computational theories in psychology like Marr's typically have broad content ascribed to their processes (*surface orientation, depth, etc.*), not narrow content.¹⁷

Egan's argument is based on an inference between computations essentially involving representational content and computations essentially involving nar-

¹⁵Egan (1995), pp. 193–194.

¹⁶Narrow content is representational content that supervenes on the intrinsic physical state of the organism possessing the content, i.e. content that a physical duplicate of the organism would have no matter what its external environment. Broad content fails to satisfy this condition: a physical duplicate of the organism could have different broad content in different environments.

¹⁷Egan (1995), pp. 194–195.

row content. Egan assumes that the only kind of content that computations could essentially involve is narrow content. In other words, that (R) entails

(R') Computation essentially involves narrow representational content

Since narrow content is either unavailable, or a positing it runs contrary to the practice of computational psychology, we should reject both (R') and (R). The cost of (R) is an unacceptable commitment to narrow content, and therefore the representation condition should be dropped.

Egan's argument works only if:

(N1) The only kind of representational content that computations could essentially involve is narrow representational content

(N2) A restriction to ascription of narrow content is intolerable in computational psychology

I will argue against (N1): there is no reason why computations in psychology, or elsewhere, cannot essentially involve broad content. Cummins (1989); Fodor (1980, 1987); Segal (1989, 1991) have argued against (N2) and for the merits of a computational psychology based purely on narrow content. I believe that we do not face such a dilemma. There is no reason why representational content in computational psychology cannot be either broad or narrow.¹⁸

What is the argument for (N1)?

One argument that Egan appears to endorse is the intuition that computational descriptions are environment-independent, and that physical duplicates are computational duplicates:

In part, the motivation for such a view [(N1)] is the recognition that computational taxonomy prescind from the subject's normal environment. Physical duplicates are computational duplicates. Given this fact, if computational states have their semantic properties essentially, then computational psychology requires a notion of content that supervenes on the physical properties of the system; in other words, it needs a notion of narrow content. (Egan, 1995, p. 194)

¹⁸Wilson (1994) argues that the vehicles of computations in psychology can extend outside an individual's head and include objects in his or her environment (cf. Clark and Chalmers (1998)). This claim should be distinguished from (N1), which concerns the content of the representations involved in computations. As noted by Segal (1997), Wilson's anti-individualism about vehicles is compatible with (N1) about content. Furthermore, Wilson's view only appears to introduce dependence on objects in the subject's nearby environment. Broad content can introduce dependence on environmental objects that are spatially and temporally distant (e.g. distant originally dubbed water samples).

However, it is hard to see how this can support (N1) in a non-question-begging way. It is unclear why someone not already convinced by (N1) would accept its premises. Why should computational descriptions be environment independent? Why should physical duplicates be computational duplicates? These are not independently plausible intuitions to someone unconvinced by the truth of (N1); they are paraphrases of (N1). The premises above are not an argument for (N1), any more than appeal to the brute intuition that *physical duplicates are intentional duplicates* constitutes an argument for individualism about intentional content. Both claims merely state individualism, they do not justify it.

A more sophisticated argument for (N1) is found in Egan (1991, 1992, 1994). This is that unless computational individuation is narrow, some important scientific generalisations are lost. According to Egan, a computational description is a description of a mechanism without reference to its environment. The environment-independence of this description allows us to make sense of a mechanism being well or ill adapted to its environment. If computational descriptions were environment-dependent, then the same computational mechanism could not be freely considered in all environments. One could not make sense of (apparently coherent) scientific claims that the same mechanism as exists in our world is well or ill adapted in other counterfactual environments.

It is precisely because a computational theory provides an environment-independent characterization of a device that we can see why *this* mechanism would not have been adaptive if the environment had been different, and why it might cease to be adaptive if the environment changes. (Egan, 1994, p. 264)¹⁹

For example, consider a subject for whom Marr's theory provides a correct account of her visual processes. Suppose this individual were transported into an environment in which her perceptual states would have a different content. If computational identity essentially depends on broad content, then the computational identity of her visual processes would change. In our environment, the computation that the subject performs is successful at recovering information from the environment. It seems coherent to ask whether, in the counterfactual environment, *the same computation* would also be successful at recovering information. But if one accepts that broad content determines computational identity, then this latter question cannot be asked. In the counterfactual environment, the same computation would not occur: the computational identity

¹⁹Also Egan (1991), pp. 199–202; Egan (1992), p. 447.

of the visual system would be different.²⁰ This appears to render incoherent apparently legitimate discussions in cognitive science about whether *the same computational system* is well or ill adapted across arbitrarily different counterfactual environments.

What can be said in response?

Egan's intuitions about the adaptation of computational mechanisms across different environments can be captured on a broad computational view with a combination of two strategies.

First, even if the computational nature of a device changes across different environments, one can still make sense of the same *physical device* being well or ill adapted to an environment. This is often all that is needed to explain intuitions in Egan's cases. One can reduce talk of whether the human visual system is well or ill adapted to an environment to discussion of whether the physical system associated with the human visual system in the actual world is well or ill adapted to that environment. If the physical system associated with the human visual system in the actual world performs poorly at recovering information in a counterfactual environment, then the computation performed by the human visual system is poorly adapted to that counterfactual environment. If the physical system performs well at recovering information from a counterfactual environment, then the computation performed by the human visual system is well adapted to that counterfactual environment. This is often all that is needed to account for our intuitions in such cases. Furthermore, one can explain why the visual system is poorly adapted to a counterfactual environment by saying that it does not perform an appropriate computation to reliably recover information about that environment. This can be true, and explanatory, even if that computation is different from the computation performed in the actual world. In short, one can accommodate claims about degree of adaptation of computational mechanisms by considering the performance of the physical system associated with the human visual system in the actual world in those counterfactual scenarios.

One can also make direct sense of talk of degree of adaption of a computation across different environments. When considering a counterfactual scenario, one can *stipulate* representational content. A physical duplicate may, in virtue of being in another environment, have a different computational identity. However, that does not stop one from *using* that physical duplicate to perform the same computation as in the actual world by setting up the appropriate representation relations. Representation relations are easy to set up; they

²⁰Note that it would be different not because the visual system might receive different input in the counterfactual case. A difference in input does not amount to a difference in a system's computational identity. The claim above is that in the counterfactual case, the nature of the *computational mechanism* that operates on inputs would be different.

can be created by stipulation. There is nothing incoherent about a physical system having more than one kind of representational content associated with it. It may have broad content, acquired in virtue of its relation to its current environment, and stipulated content, acquired in virtue of the suppositions under which we imagine the counterfactual scenario. Claims about how a computation in the actual world performs in counterfactual environments can be understood as claims about the performance of a physical duplicate in the counterfactual environment where that system is interpreted as performing the same computation (i.e. where we set up the appropriate representational conventions as part of our counterfactual imagining). In other words, the question we entertain is: *supposing* that a physical duplicate were to perform the same computation in the counterfactual environment, how successful would it be in that environment?

When thinking about the degree of adaptation of a computational process across different environments, I think that we vacillate between these two strategies. Sometimes we acknowledge that in the counterfactual environment the physical system performs a different computation in virtue of having different representational content (as strategy 1). Sometimes we imagine the physical system in such a way as to force it to perform the same computation regardless of its environment (as strategy 2).

The cases discussed by Burge (1986) bring the dual nature of our intuitions about computational identity to the fore. Burge describes the human visual system as performing an adaptive *crack*-processing computation in one (crack-based) environment, and a physical duplicate of the organism as performing an adaptive *shadow*-processing computation in another (shadow-based) environment.²¹ The physical duplicates have different broad content, and on Burge's view perform different computations. The former embodies assumptions about *cracks*. The latter embodies assumptions about *shadows*. We can make sense of the following two intuitions. First, we can make sense of the intuition that *the computation performed by the human visual system is adaptive* in both environments (under strategy 1). Second, we can see each system as performing the same computation by representational stipulation—this allows us to make sense of the intuition that the former computation would embody false assumptions about shadows, and hence be a poor way of detecting shadows in its environment; while the latter computation would embody false assumptions about cracks, and hence be a poor way of detecting cracks in its environment. Switching one's attention between the computation determined by broad content, and the computation determined by representational stipulation, seems to be part and parcel of discourse about computations in counterfactual situations.

²¹Burge (1986), pp. 39–43.

The combination of the two strategies allows one to accommodate the dual thoughts that in a sense, the computation performed by the human visual system is adaptive in both environments, and in another, that the computation performed in each environment is adaptive only in its own environment. Both intuitions appear legitimate and compelling; one or the other tends to dominate in different contexts. The strategies above seem better able to accommodate the dual nature of our intuitions about computational identity across counterfactual environments than a restriction to narrow individuation. The two strategies also show that one is not forced to accept (N1) in order to make sense of claims about the adaption of computations across different environments.

The third argument for (N1) derives from Fodor (1980). It involves what Fodor calls the *formality condition*. According to Fodor, there is something about the nature of computation that restricts computations in psychology, and indeed any type of computation, to narrow content. Fodor's exact argument is hard to pin down. I will try to develop it below.

The motivation for the formality condition is to make sense of the ability of computations, discussed in Section 2.1, to appear to be semantically-sensitive, without spookily having access to what their states refer to. Computations are formal, and therefore their transitions are governed by the syntactic character or 'shape' of their states. However, a computation can appear to be semantically-sensitive if the formal character of its states covaries with its representational content. By tracking its formal properties, a computation can appear to track semantic properties. This suggests a formality condition on computation. Any computation that appears to be semantically-sensitive should have all relevant semantic distinctions mirrored in formal differences among its tokens. So, for example:

the computational theory of mind requires that two thoughts can be distinct in content only if they can be identified with relations to formally distinct representations. (Fodor, 1980, p. 486)

The form of computational states should mirror their representational content: different representational content must be encoded by different forms. The problem is that broad content dramatically fails to satisfy this condition. States in two physically identical individuals can be physically and formally type-identical, they can be physical duplicates, and yet have different broad contents. Narrow content appears to be the only type of representational content that can hope to satisfy the formality condition:

Narrow psychological states are those individuated in light of the formality condition; viz., without reference to such semantic properties as truth and reference. And honoring the formality condition

is part and parcel of [computational psychology]. (Fodor, 1980, p. 495)

The problem that broad content raises for computation is that it appears to introduce the kind of spooky dependence on referents that the notion of computation was intended to eliminate. Suppose one individuates one's computational tokens in terms of their broad content. If computations are individuated in terms of representational content, then at a minimum, a computation should be consistent in how it handles that representational content—it should, for example, consistently map the same representational content to the same representational content. If computations cannot consistently handle certain kinds of representational content, then it makes no sense to individuate those computations in terms of how they handle that kind of representational content. But how can a computation consistently handle broad content without being spookily sensitive to its referent? How can it, say, know that some of its tokens have the broad content *water*, while others have the broad content *twater* and should be processed in different way, when the tokens are formally indistinguishable? The only plausible kind content by which to individuate computations—the only kind that computations can consistently process without spooky knowledge of their referent—is narrow content. Therefore, the only type of content relevant to the individuation of computations, the only kind of content they can essentially involve, is narrow content.

What can be said for this argument for (N1)?

First, one should clarify that the basic units of computation are processes, not states. A state counts as computational only to the extent that it participates in a computational process. It makes no sense to posit a computational state in isolation. Computational states must have 'owners', associated computational processes of which they are part (just as digestion states must have 'owners', associated digestion processes of which they are part). There is no utility in the notion of an isolated computational state. Computational states are also not individuated in isolation. They are individuated in part by reference to the computational process in which they occur. A 5 V signal in one machine may play the role of a halting state, and 0 V the role of a starting state, while a physically identical 5 V in another machine may play the role of a starting state, and 0 V the role of a halting state. One cannot identify a state as a starting or halting state independently of the process in which it occurs. The computational identity of a state depends on the process in which it participates. Just as vocalisations are individuated as distinct words only relative to a language of which they are part, so physical states are typed as computational only relative to the computational process(es) of which they are part. A computational state

must have an associated 'owner' computational process.

The motivation for the formality condition is to make sense of the apparent semantically-sensitivity of computations. What is the minimum required to explain this? All that is required is that the relevant semantic distinctions between a *process's* states should be mirrored in formal differences among those states. It is not required that all semantic properties, or semantic differences between the states of that computational process and those of other computational processes, should be so marked. Only the semantic differences to which a computational process appears to be sensitive need to be marked formally by its states.

For example, suppose a computation appears to distinguish input tokens in English that represent *fire* from those that represent *water*. If presented with an input token that represents *fire*, the computation outputs the string of characters 'fire is hot', and if presented with an input token that represents *water*, it outputs the string 'water is wet'. Such a computation is a simple example of apparent semantic sensitivity. How could such a computation work? Unless it is spookily sensitive to its referents, it works by the relevant semantic differences between its tokens being mirrored in formal differences. The input tokens must have some formal difference that consistently distinguishes tokens that represent *fire* from those that represent *water*. Just one formal difference is required; the tokens need not encode any finer grained differences than that required to explain the behaviour above. For example, it is not necessary that a formal difference between tokens that represent *fire* and *chalk* be marked to explain the sensitivity above. More accurately, the formal structure of the tokens need only encode the information that *fire* and *water* tokens are relevantly *different* tokens. It need not encode all their semantic properties. In particular, it need not determine their referents: it need not determine that *fire* tokens refer to *fire*, and *water* tokens refer to *water*, and rule out that the respective tokens refer to, say, *0* and *1*, or *chalk* and *cheese*, or any other pair of distinct contents.

There is no reason, stemming from explanation of apparent semantic sensitivity, to think that the formal structure of a computational state should determine its representational content. In short, there is no reason why its representational content should supervene on its formal properties. The formal structure is only needed for the computation to keep track of the semantic differences relevant to the process, not to encode all semantic properties. Consequently, there is no reason why the contents of computational tokens cannot be broad content. So long as a difference between the *fire* and *water* tokens is marked, it does not matter whether their content is broad or narrow.

One might argue that broad content is still problematic. If computations are individuated in terms of representational content, then a computation should

be consistent in how it processes representational content—it should map the same representational content to the same representational content. One might think that broad content cannot satisfy this condition. Two tokens can have different broad content and yet be physically identical. Such tokens appear apt to disrupt the consistent processing of representational content. What is to stop a computation that ostensibly sets up a *water–water* relationship from, on any given occasion, producing a spurious *water–twater* relationship, or any other *water–X* relationships, where *X* is physically identical to a *water* token but with different broad content? The computation cannot distinguish between *water*, *twater*, and *X* tokens, so there seems no way in which it can consistently respond to, and yield, all and only *water* tokens. Any *water* processing computation appears vulnerable to the introduction of doppelgangers with different broad content. Therefore, computations cannot establish consistent relationships between broad content, and the processing of broad content cannot be a consistent way of individuating computations.

Note that the relevant cases of broad content all involve environment dependency. Two physically identical individuals have different content (*water* and *twater*) because they exist in different environments (Earth and Twin Earth). Within a single environment, broad content is univocal. The objection above concerns whether a computation can consistently process tokens with broad content. Any given computational process occurs within an environment, and throughout that process, its environment fixes univocal broad content.²² The computational process hypothesised above, which simultaneously contains a mixture of physically identical *water* and *twater* tokens, is an incoherent possibility. Tokens within a process are either all *water* tokens or all *twater* tokens depending on the environment of the process. The comparison relevant for whether broad content can be consistently processed is with tokens that could take place *in the same process*, not tokens that could take place anywhere in any environment. Within an environment, and a computation always occurs within an environment, there is no reason why computations cannot process broad content consistently. Broad content may vary between physical duplicates in different environments. But this kind of variation does not threaten the consistent handling of representations by individual computational processes.

²²Cases where the computational process is transported between environments during the computation, or the environment changes during the computation, cause no serious problems. These should be treated in the same way as those of individuals transported from Earth to Twin Earth. The general intuition here is that the individual would continue having Earthly content for some time after the transportation (certainly enough to complete any thoughts), and only gradually acquire Twin Earth broad content as he or she becomes embedded in Twin Earth representation relations and conventions (Block, 1990). There is little danger of inconsistent content or information processing. Cases in which a process is so large as to span two environments pose no problem either, since the process can be consistent in how it handles broad content within each portion of the process in the different environments, and this is sufficient for consistent individuation.

What is wrong with saying that my physically identical Twin Earth counterpart possesses an information-processing subsystem that reliably processes *twater* tokens, while I possess a reliable *water* processor? There is no reason why a computation in an environment cannot process broad content just as consistently as it can narrow content.

While Fodor's formality condition on computation is:

- (F1) Any difference in representational content between computational states should be mirrored in a difference in their formal structure.

A weaker condition suffices to make sense of apparent semantic-sensitivity:

- (F2) For a computational process, *P*, all differences in representational content to which the *P* appears to be sensitive should be mirrored in a formal difference between *P*'s states.

(F2) is compatible with the representational content to which a computation appears to be sensitive being broad or narrow. Therefore, there is no argument from the formality condition to (N1).

Why does Fodor endorse (F1) rather than (F2)? When explaining semantic sensitivity, the comparison class one has in mind is crucial. If the comparison class contains all possible computational states, then every aspect of representational content should supervene on formal structure and (F1) follows. However, as should be clear, the correct comparison class is smaller: the class of other tokens that could occur in the computational state's 'owner' process. If the associations described above between computational states and processes are ignored, then incorrect comparisons are made between states in one process (or one environment) and those in another. With those connections in mind, (F1) can be replaced with (F2) with no loss to explanations of apparent semantic sensitivity.

A wider consequence is that there are no interesting constraints on whether computational psychology is individualistic from the nature of computation. *Contra* Fodor (and Egan), computational psychology does not need to be individualistic. This raises the question: given that there are no principled constraints from the nature of computation, are there any constraints from the practice of computational psychology as to whether computational psychology should be individualistic? Egan (1991) argues that when this question is asked about psychology as a whole, there is no single correct answer. In some cases, psychological explanations should attribute narrow content, such as Marr's ascription of representations of proximal features of subject's visual image, e.g. *blobs*, *virtual lines*, and *zero-crossings*. In other cases, psychological

explanations should attribute broad content, such as Marr’s ascription of representations of distal features in the environment, e.g. *surfaces*, *shadows*, and *cracks*. A mixed answer should be expected to a question about individualism for psychology as a whole. On the view described above, one should expect a similarly mixed answer to the question of whether computational psychology is individualistic. There is no reason why computational psychology should be restricted to narrow content.

3 Arguments for the representation condition

The primary purpose of this paper is to show that Egan’s attack on the representation condition does not succeed. But it is also worth briefly considering some of the positive arguments for (R). In Section 2.2, I argued that there are reasons internal to Egan’s position that should force her to make essential use of representation. I now want to turn to some of the reasons, independent of the details of Egan’s view, for why one should accept (R). I will only attempt to outline the arguments that I think should underlie a defence of (R).

A computation maps certain inputs to certain outputs. A physical computation is a mapping between real-world *stuff*: it takes stuff (ink-marks, electrical signals, etc.) as input and yields other stuff, or other arrangements of stuff, as output. The claim defended below is that computations are not just mappings between any kind of stuff, but mappings between *stuff that represents*.

This claim does not place any restriction on the type of representation relation involved. Nothing is said, for example, to require that the inputs and outputs of computational processes must *intrinsically* or *naturally* represent. In some cases this may be true—the inputs and outputs may naturally represent—in other cases it may not—the inputs and outputs only represent because we, as humans, interpret them as doing so. Nevertheless, whatever kinds of facts underlie representation, the inputs and outputs of a computation must represent.

Here are three arguments for why computation has to involve representational content.

3.1 Paradigmatic cases of computation involve representation.

Many paradigmatic cases of computation involve representation. For example, Turing’s clerk, who performs computations by hand, performs a mapping between representations.²³ The clerk maps representations (ink-marks on the

²³For a description of why human computers are paradigmatic cases of computation, see Gandy (1988); Sieg (1994, 2001).

page) to other representations (other ink-marks on the page). The clerk's ink-marks can be interpreted as representing either numerals or numbers. This ambiguity in representational content is not unusual. The following ink-marks, 1, can represent either the numeral '1' or the number 1, depending on context. The context can be partially specified by adding quotation marks. However, this convention is not always decisive. In many cases there is opportunity to interpret the ink-marks either way.²⁴ It is not unusual for the same physical stuff to have multiple representational contents associated with it, and consequently for the same physical process to have multiple computational identities.

Another paradigmatic case, electronic computation, also involves representation. An electronic computer takes electrical signals as input and yields electrical signals as output. The input and output electrical signals of a computer are not just any electrical signals, but electrical signals that represent. Typically, the electrical signals of a computer represent 0's and 1's. Again, there is possibility of multiple representation. A given signal may represent *both* a long sequence of 0's and 1's, *and* the text of a new e-mail message. Or, a given signal may represent *both* a sequence of 0's and 1's, *and* a picture to display on the screen.

The value of paradigmatic cases of computation involving representation is far from conclusive as an argument for (R), but it is not negligible. When questions arise as to whether certain borderline cases count as computations or not, it is salutary to keep the paradigmatic cases in mind. The presence of representation in all the paradigmatic cases of computation is data to be taken under consideration. At the least, it demonstrates a marked lack of paradigmatic cases of computation without representational content.

3.2 Representation is involved in the notion of I/O equivalence.

Our notion of computation includes a notion of input–output (I/O) equivalence. We often claim that diverse physical systems are 'computationally equivalent' or 'compute the same function'. What we mean by this is that the two systems perform the same computational task, even if they use different methods to achieve it; in other words, they are I/O equivalent. I/O equivalence is a necessary, but not sufficient, condition for computational identity. There is more to the notion of computational identity than I/O equivalence, but I/O equivalence is an essential component of that notion. I wish to argue that it is hard to make sense of I/O equivalence without assuming that computation involves representational content.

²⁴That there is an ambiguity about *content* here seems required to make sense of the use/mention confusion. What is at stake in a disagreement about use/mention is whether one is talking *about* numerals or numbers when employing certain ink-marks.

Imagine two I/O equivalent systems that are made out of different materials. One system may be made out of silicon and take electrical signals as inputs and outputs, the other system may be made out of tin-cans and string and take marbles as inputs and outputs. Suppose that the two systems are I/O equivalent. What could their I/O equivalence consist in? The respective inputs and outputs of the two systems may be so different as to not have any physical, structural, or functional properties in common. The only plausible answer seems to be that their respective inputs and outputs *represent the same thing*.

Another example would be two computational systems that perform the same numerical calculation. Suppose that one system takes ink-marks shaped like Roman numerals (I, II, III, IV, ...) as input and yields ink-marks shaped like Roman numerals as output. Suppose that the other system takes ink-marks shaped like Arabic numerals (1, 2, 3, 4, ...) as input and yields ink-marks shaped like Arabic numerals as output. Suppose that we wish to say that the two systems are I/O equivalent. What could their I/O equivalence consist in? There need be no physical or functional similarity between their respective inputs and outputs. The only respect in which the two systems are I/O equivalent seems to be that their inputs and outputs represent the same thing.

Egan claims that the inputs and outputs of computations can be characterised in a purely functional way that does not appeal to representational content:

To describe something as a symbol is to imply that it is semantically interpretable, but (and this is the important point) its type identity as a symbol is independent of any particular semantic interpretation it might have. Symbols are just functionally characterized objects whose individuation conditions are specified by a realization function f_R . (Egan, 1992, p. 446)

Similarly, Piccinini claims that one can make sense of facts about the I/O equivalence of computations using functionally characterised symbols and without appeal to their representational content.²⁵ The problem is that there do not seem to be sufficiently broad non-semantic functional characterisations to capture all the relevant facts about computation and computational individuation. Given the huge diversity of physical and functional properties of systems that we wish to judge as 'computationally equivalent', restricting attention to non-semantic resources cannot appear to account for our computation talk. The only thing that two physically diverse inputs and outputs have in common is that they represent the same thing.

²⁵Piccinini (2008), pp. 223–224.

3.3 Representation is needed to make some basic distinctions.²⁶

Any plausible notion of computation needs to be able to make certain basic distinctions. One such distinction is that between AND gates and OR gates—the building blocks of many electronic computers. AND and OR gates have the following characteristics. The output of an AND gate is 1 just in case both inputs are 1 , otherwise it is 0 . The output of an OR gate is 0 just in case both inputs are 0 , otherwise it is 1 .

a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

Table 1: AND and OR gates

Consider an electrical system with following characteristics. The system gives an output of 5 V if both its inputs are 5 V, otherwise it gives an output of 0 V. Does this system implement an AND gate or an OR gate? At first glance, the system appears to implement an AND gate: it gives an output with 5 V just in case both its first *and* its second inputs are 5 V. But why should 5 V be associated with 1 , and 0 V with 0 , rather than the other way around? If 5 V is associated with 0 , and 0 V with 1 , then according to the tables above, the system implements an OR gate. So which gate does the system implement? As the system has been described so far, there is nothing to decide between the two options. No physical or structural property decides whether 5 V should be paired with 1 , and 0 V with 0 , or 5 V with 0 , and 0 V with 1 . The situation is symmetrical with respect to both assignments.

in_1	in_2	out
0 V	0 V	0 V
0 V	5 V	0 V
5 V	0 V	0 V
5 V	5 V	5 V

Table 2: An implementation of an AND gate or an OR gate?

The notion of representation allows us to decide between these two options. We can say that *if* an electrical signal of 5 V represents 1 , and *if* an electrical

²⁶This argument is related to Shagrir (2001), which warrants fuller discussion. Briefly, unlike Shagrir’s argument, it uses a simpler physical system and argues more directly for the role of content in computational individuation. It does not rely on a notion of a ‘maximal task’ (objected to by Piccinini (2008), p. 229). It also places representation at the centre of computational individuation rather than a final, and arguably optional, constraint (Piccinini (2008), p. 230).

signal of 0 V represents 0, then the system implements an AND gate. Alternatively, if an electrical signal of 0 V represents 1, and if an electrical signal of 5 V represents 0, then the system implements an OR gate. The difference between an implementation of an AND gate and an OR gate is a difference in representational content.

The representational nature of real-world computation is sometimes obscured by the widely accepted claim that computation is *syntactic*. Computation is syntactic in at least two senses. First, as we saw in Section 2.3, computation is sensitive to the formal, syntactic, structure of its input. This requirement is, of course, compatible with the claim that such input has representational content. The second sense in which computation is syntactic is that the inputs and outputs of a computation often *represent* syntactic entities. We often take an input to a computational process to represent a numeral ('0' or '1') rather than a number (0 or 1). Thus, one often finds the inputs and outputs of an AND gate labelled with the numeral '0' or '1', and this numeral called its 'syntactic content'. Such content may be syntactic, but it is representational nevertheless.

Another possible source of confusion about syntax arises from the conflation of the notion of real-world computation with the notion of computation in mathematics. As argued in Section 2.2, real-world computation and computation in mathematical computation theory are different. A Turing machine employs the mathematical notion of computation. A Turing machine is an abstract mathematical object; it does not 'perform' a computation in the same way as a physical system. A Turing machine is typically identified with a set of sets of symbols. It is not dissimilar in ontological status to a mathematical function, such as $f(x) = x^2$. A Turing machine, at least in the first instance, operates on numerals instead of numbers. It takes numerals (symbols) as input and yields numerals (symbols) as output. A Turing machine therefore operates on syntactic entities.²⁷

Two points should be made about these syntactic entities. First, syntactic entities such as numerals are themselves abstract objects—they are not identical to ink-marks on the page, although ink-marks may represent them. Second, syntactic entities are commonly thought of as uninterpreted in this context, i.e. as lacking representational content. Hence, mathematical computation does not need to operate on entities that represent. This may lead one to think that the computations performed by real-world systems do not need to operate on entities that represent either. Unfortunately, this is not true. Although symbolic entities, such as numerals, provide a way of individuating Turing machines,

²⁷See Boolos et al. (2002), pp. 24–25, for more on this point.

these objects are not available in the real-world. In real-world computation, we are stuck with physical stuff, such as ink-marks and electrical impulses. As we have seen, the only way for such stuff to support a plausible notion of computational identity is to employ the notion of representation. The non-representational nature of mathematical computation does not transfer over to real-world computation. The two notions of computation have different problems and commitments.

4 Conclusion

The questions with which we started were the questions that motivate a philosophical theory of computation. What is the difference between a computation and any other physical process? Under what conditions are two computations the same or different? Partial answers to these questions are now available. A computation essentially involves the manipulation of representations; computations are consistent ways of mapping representational content to representational content. This is not true of all physical processes. Two physical systems perform the same computation only if they map the same representational content to the same representational content. I/O equivalence, understood in terms of content, is a necessary condition for computational identity. These answers are partial for two reasons. First, not every process that maps representations in this way is a computation. A computation should, in a sense to be defined, be mechanical: there should be a comprehensible, counterfactually robust, inter-linked, series of steps in how it achieves its mapping between representational content. Second, as noted above, I/O equivalence is only a necessary condition on computational identity. Two physical systems perform the same computation if and only if they are I/O equivalent and they achieve their mapping between representational content *in the same way*. Explicating this latter notion requires considering whether the inter-linked series of steps are appropriately equivalent. There is opportunity here for representational content to enter into the story again. However, a detailed treatment of this condition, the subject matter of a full-blown positive account of computation, which has to wait for another occasion. For the moment, it should be clear that the representation condition is a reasonable condition on computation, and that Egan's arguments against it do not hold up.

References

- Bechtel, W. and Richardson, R. C. (1993). *Discovering Complexity: Decomposition and Localization as Scientific Research Strategies*. Princeton University Press, Princeton, NJ.

- Block, N. (1990). Inverted Earth. *Philosophical Perspectives*, 4:53–79.
- Boolos, G., Burgess, J. P., and Jeffrey, R. C. (2002). *Computability and Logic*. Cambridge University Press, Cambridge, 4th edition.
- Burge, T. (1986). Individualism and psychology. *Philosophical Review*, 95:3–45.
- Chalmers, D. J. (1996). Does a rock implement every finite-state automaton. *Synthese*, 108:309–333.
- Churchland, P. S. (1986). *Neurophilosophy*. MIT Press, Cambridge, MA.
- Clark, A. and Chalmers, D. J. (1998). The extended mind. *Analysis*, 58:7–19.
- Crane, T. (2003). *The Mechanical Mind*. Routledge, London, 2nd edition.
- Craver, C. F. (2007). *Explaining the Brain*. Oxford University Press, Oxford.
- Cummins, R. (1989). *Meaning and Mental Representation*. MIT Press, Cambridge, MA.
- Davies, M. (1991). Individualism and perceptual content. *Mind*, 100:461–484.
- Dennett, D. C. (1971). Intentional systems. *Journal of Philosophy*, 68:87–106.
- Dennett, D. C. (1987). *The Intentional Stance*. MIT Press, Cambridge, MA.
- Egan, F. (1991). Must psychology be individualistic? *Philosophical Review*, 100:179–203.
- Egan, F. (1992). Individualism, computation, and perceptual content. *Mind*, 101:443–459.
- Egan, F. (1994). Individualism and vision theory. *Analysis*, 54:258–264.
- Egan, F. (1995). Computation and content. *Philosophical Review*, 104:181–204.
- Fodor, J. A. (1980). Methodological solipsism considered as a research strategy in cognitive psychology. *Behavioral and Brain Sciences*, 3:63–109. (Reprinted in D. M. Rosenthal, editor, *The Nature of Mind*).
- Fodor, J. A. (1987). *Psychosemantics*. MIT Press, Cambridge, MA.
- Fodor, J. A. (1998). *Concepts*. Blackwell, Oxford.
- Gandy, R. O. (1988). The confluence of ideas in 1936. In Herken, R., editor, *The Universal Turing Machine: A Half-Century Survey*, pages 55–111. Oxford University Press, Oxford.
- Glennan, S. (2002). Rethinking mechanistic explanation. *Philosophy of Science*, 69:S342–S353.
- Kitcher, P. (1988). Marr’s computational theory of vision. *Philosophy of Science*, 55:1–24.
- Machamer, P. K., Darden, L., and Craver, C. F. (2000). Thinking about mechanisms. *Philosophy of Science*, 67:1–25.

- Marr, D. (1982). *Vision*. W. H. Freeman, San Francisco, CA.
- Morton, P. (1993). Supervenience and computational explanation in vision theory. *Philosophy of Science*, 60:86–99.
- Newell, A. (1982). The knowledge level. *Artificial Intelligence*, 18:87–127.
- Piccinini, G. (2008). Computation without representation. *Philosophical Studies*, 137:205–241.
- Putnam, H. (1988). *Representation and Reality*. MIT Press, Cambridge, MA.
- Pylyshyn, Z. W. (1984). *Computation and Cognition*. MIT Press, Cambridge, MA.
- Searle, J. R. (1990). Is the brain's mind a computer program? *Scientific American*, 262:20–25.
- Searle, J. R. (1992). *The Rediscovery of the Mind*. MIT Press, Cambridge, MA.
- Segal, G. (1989). On seeing what is not there. *Philosophical Review*, 98:189–214.
- Segal, G. (1991). Defence of a reasonable individualism. *Mind*, 100:485–493.
- Segal, G. (1997). Review of Wilson, R. A., *Cartesian Psychology and Physical Minds*. *British Journal for the Philosophy of Science*, 48:151–156.
- Shagrir, O. (2001). Content, computation and externalism. *Mind*, 110:369–400.
- Shapiro, L. (1997). A clearer vision. *Philosophy of Science*, 64:131–153.
- Sieg, W. (1994). Mechanical procedures and mathematical experience. In George, A., editor, *Mathematics and Mind*, pages 71–117. Oxford University Press, Oxford.
- Sieg, W. (2001). Calculation by man and machine: Conceptual analysis. In Sieg, W., Sommer, R., and Talcot, C., editors, *Reflections on the Foundations of Mathematics (Essays in Honor of Solomon Feferman)*, pages 387–406. Volume 15 of Lectures Notes in Logic, Association of Symbolic Logic.
- Stich, S. P. (1983). *From Folk Psychology to Cognitive Science*. MIT Press, Cambridge, MA.
- Sudkamp, T. A. (1998). *Languages and Machines*. Addison-Wesley, Reading, MA, 2nd edition.
- Wilson, R. A. (1994). Wide computationalism. *Mind*, 103:351–372.