Synthese (2022) 200:189
https://doi.org/10.1007/s11229-022-03691-1

ORIGINAL RESEARCH

Structuralism, indiscernibility, and physical computation

F. T. Doherty[1] · J. Dewhurst[2]

Received: 31 March 2021 / Accepted: 31 March 2022 / Published online: 27 April 2022

Abstract
Structuralism about mathematical objects and structuralist accounts of physical computation both face indeterminacy objections. For the former, the problem arises for cases such as the complex roots i and $-i$, for which a (non-trivial) automorphism can be defined, thus establishing the structural identity of these importantly distinct mathematical objects (see e.g. Keränen in Philos Math 3:308–330, 2001). In the case of the latter, the problem arises for logical duals such as AND and OR, which have invertible structural profiles (see e.g. Shagrir in Mind 110(438):369–400, 2001). This makes their physical implementations indeterminate, in the sense that their structural profiles alone cannot establish whether a given physical component is an AND-gate or an OR-gate. Doherty (PhilPapers, https://philpapers.org/rec/DOHCI-3, 2021) has recently shown both problems to be analogous, and has argued that computational structuralism is threatened with the absurd conclusion that computational *digits* might be indiscernible, such that, if structural properties are all that we have to go on, the binary digit 0 must be treated as identical to the binary digit 1 (rendering pure structuralism absurd). However, we think that a solution to the indiscernibility problem for mathematical structuralists, drawing on the work of David Hilbert, can be adapted for the analogous problem in the computational case, thereby rescuing the structuralist approach to physical computation.

Keywords Structuralism · Computation · Indeterminacy · Indiscernibility · Hilbert

Introduction

Structuralist accounts of physical computation are those that ground computational identity in local physical structure, such as classical causal accounts and more recent mechanistic accounts. One particular line of objection to such accounts has received

✉ J. Dewhurst
joseph.e.dewhurst@gmail.com

1 Independent scholar, Glasgow, Scotland

2 LMU Munich, Munich, Germany

substantial attention in the literature, namely that structuralists cannot provide an adequate account of basic computational individuation, i.e., an account of which (physical) component implements which (abstract) logical function. Critics point to the structural indeterminacy of logical duals, such as OR and AND, to demonstrate that structure alone cannot successfully individuate computational states and processes. Additional ingredients, such as semantic content, must therefore be required to provide an adequate account of computation (Sprevak, 2010; Shagrir, 2001, 2018).

In response, computational structuralists have, for the most part, embraced such indeterminacy as consistent with their broader structuralist thesis, rejecting the semanticist's assumption that there is always a fact of the matter about which physical component implements which logical function (see e.g. Piccinini, 2015; Dewhurst, 2018a). However, Doherty (2021) has recently shown that the *indeterminacy* objection to computational structuralism is actually symptomatic of an underlying and far more serious *indiscernibility* problem, according to which structuralists are at risk of counting logical duals (and even binary digits) not only as indeterminate but in fact as *identical*—a presumably absurd conclusion that even the structuralist would wish to avoid.

Given Doherty's demonstration that functional indeterminacy implies structural identity, structuralists are no longer able to bite the bullet regarding functional indeterminacy, and a new defence of computational structuralism is therefore required. In this paper we provide such a defence by drawing on an analogous debate in the philosophy of mathematics and the early mathematical structuralism of David Hilbert. We thereby rescue computational structuralism, not only from the absurd indiscernibility that Doherty claims to reduce it to, but also from the original indeterminacy objection. The 'Hilbertian computational structuralism' that results from our analysis is able to bypass concerns about indiscernability or indeterminacy, but might still imply a form of pancomputationalism, which we do not address directly here and aim to return to in future work. Our analysis lays the groundwork for a pure structuralist account of computational individuation, and also indicates some interesting parallels between the philosophy of computation and the philosophy of mathematics.

In Sect. 1 we review computational structuralism and the indeterminacy objection, before presenting Doherty's new challenge to computational structuralism in Sect. 2. In Sect. 3 we introduce the Hilbertian response to an analogous challenge to mathematical structuralism. In Sect. 4 we apply this response to the specific case of computational structuralism, including a discussion of the practical implications of these arguments.

1 Computational structuralism and the indeterminacy objection

In this section we will introduce structuralist accounts of physical computation and the indeterminacy objections that have recently been raised against them. Those familiar with the details of these debates can skip to Sect. 2.

1.1 Computational structuralism

Structuralism about computational implementation (or 'computational structuralism') is the view that what matters for determining the identity of (some part of) a physically implemented computational system is just the local causal structure of the system in question, or more precisely, some kind of mapping between that structure and an abstract (formal) computational structure (cf. Rescorla, 2013). Classic examples of this view include Egan (1992), Chalmers (1996), and Copeland (1996), but it can also be found in a more contemporary guise in the mechanistic accounts of computation due to Piccinini (2007, 2015), Miłkowski (2013), and Fresco (2015).[1] The opposing semantic (or representational) view of computational implementation holds that, in addition to possessing the right kind of causal structure, a physically implemented computational system must also possess semantic (or representational) content. The classic example of this view is Fodor's (in)famous dictum that there can be "no computation without representation" (Fodor, 1998, p. 34), but other more recent examples include Shagrir (2001, 2018), Sprevak (2010), and Rescorla (2013, 2014).

Of course, mappings are cheap, and so while the computational structuralist claims that a physical system implements an abstract computation if a mapping can be identified between the causal structure of the physical system and the formal structure of the abstract computation, this requirement is typically buttressed with additional constraints. Candidate constraints include, for example: possessing the right kind of causal structure (Chalmers, 1996); having the (proper) function of computing (Piccinini, 2015); or identifying a sufficiently simple mapping (Millhouse, 2019). None of these constraints has yet proven to be entirely successful, and producing novel constraints (or objections to existing constraints) now constitutes a small cottage industry. However, the problem that we are concerned with here does not directly target the mapping component of computational structuralism, so we will assume for the time being that some appropriate constraint can be found. A physical computation would then just be a causal process whose structure in some sense mirrors the formal (logical) structure of an abstract computation.[2]

Consider a simple computational component, a logic gate (or processor) with the following physical profile: The gate takes two voltage levels as input, and produces a single voltage level as output. For the sake of simplicity, let's say that the gate is sensitive only to low (0V) or high (5V) voltage levels and gives a 5V output if, and only if, it receives two 5V inputs, otherwise giving a 0V output (see Table 1).

This table is potentially misleading in an important way: most views—and certainly structuralist views—don't take *all* of the properties of the physical states, such as precise voltage levels, to be computationally relevant. Rather it is the distinction *between* voltage levels that is relevant for computation. The computationally relevant

[1] Insofar as they are reliant on some notion of 'proper' (or teleological) functions, there is a question mark about how *purely* structuralist mechanistic accounts of computation really are, but see Dewhurst (2018b) for an attempt to formulate a version of the mechanistic account that does not rely on proper functions.

[2] Semanticists such as Sprevak and Shagrir would also agree that physical computation involves a mirroring relationship of this kind, but would deny that such a relationship is *sufficient* for a non-semantic system to qualify as computational.

Table 1 Gate 1's physical profile

Input 1	Input 2	Output
5V	5V	5V
5V	0V	0V
0V	5V	0V
0V	0V	0V

Table 2 Gate 1's structural profile

Input 1	Input 2	Output
1	1	1
1	0	0
0	1	0
0	0	0

properties of the above physical states, i.e. that they are distinct, are conventionally indicated by Boolean values, as depicted in Table 2 above.

Here 0/1 indicates only the computationally relevant properties of the physical states 0V/5V. For computational structuralists these properties will consist in the causal structure of the physical states within the context of the system. We stress this point because the status of 0/1 will be important later on. Henceforth we will refer to 0/1 as computational *digits*, understood as above. Note that although digits are in some sense slight abstractions from the physical states, they should not be understood as purely abstract states (like truth values), but rather indicate that we are only interested in the computationally relevant features of the physical states implementing the computation. What matters about our inputs and outputs is not their voltage level *qua* voltages, but rather that the processor systematically recognises two distinct component types, and the Boolean notation can help us to make this clear. The computation carried out would be the same regardless of the physical details of these component types, provided that the processor could systematically distinguish them, but they are nonetheless physical components. This is just to say that what matters for physical computation are certain medium-independent properties of the physical system in question, which, according to the structuralist, are best characterised as structural properties. In this sense 0V/5V is also sometimes used with equal legitimacy to indicate the computationally relevant features of the voltage levels. We simply want to be clear on the distinction between the physical states themselves and their computationally relevant aspects (in this case, some of their structural properties), and so we choose to use Boolean values to highlight this (sometimes ignored) distinction.

At any rate, a processor of this kind is conventionally treated as implementing the (abstract) logical function AND by mapping 0 to FALSE (or F) and 1 to TRUE (or T), as in Table 3 below.

Another physical component, with the same inputs but outputting 1 (5V) unless both inputs are 0 (0V), is conventionally treated as implementing OR, by carrying out the same mappings as above (see Tables 4, 5, and 6).

Table 3 Gate 1 interpreted as AND

Input 1	Input 2	Output
T	T	T
T	F	F
F	T	F
F	F	F

Table 4 Gate 2's physical profile

Input 1	Input 2	Output
5V	5V	5V
5V	0V	5V
0V	5V	5V
0V	0V	0V

Table 5 Gate 2's structural profile

Input 1	Input 2	Output
1	1	1
1	0	1
0	1	1
0	0	0

Table 6 Gate 2 interpreted as OR

Input 1	Input 2	Output
T	T	T
T	F	T
F	T	T
F	F	F

So far so good: we have two (physical) computational components corresponding to two different (abstract) logical functions. This approach can be extended to other logical functions/computational components, in principle giving a full account of computational implementation (see, e.g. Piccinini, 2015 for more detail). A structuralist approach to computational implementation could offer a fully naturalistic foundation for physical computation, one that both avoids unlimited pancomputationalism and also avoids making any appeal to metaphysically suspect notions such as semantic content or, in the case of pure structuralism, proper functions.[3] If a structuralist account were successful then it could in principle be used in turn to ground such notions – this is the approach taken by previous mechanists such as Milkowski and Piccinini, who

[3] Dewhurst (2018b) has already defended a version of the mechanistic account (which we take to be a kind of computational structuralism) that, unlike Piccinini's classic mechanistic account, does not make any appeal to proper functions. An account of this kind is what we mean by 'pure structuralism', i.e. a structuralism that avoids making any appeal to intrinsic, non-structural properties of computational objects, such as semantic content or proper functions.

allow that a non-semantic account of physical computation could be used to support a downstream theory of semantic content. Alternatively, one could retain an entirely non-semantic account of physical computation, and insist that there is no close connection between computation and representation. Either option would be consistent with the foundation provided by computational structuralism, and the account itself does not adjudicate between further questions about whether, for example, computational theories of *cognition* should be understood as representational. Computational structuralism would therefore provide a general account of physical computation that could remain neutral with regard to its subsequent applications. However, it is vulnerable to a now well-know objection to do with indeterminacy, to which we turn next.

1.2 The indeterminacy objection

The indeterminacy objection to computational structuralism was first raised by Shagrir (2001) and then, in a simplified form, by Sprevak (2010). Both Shagrir and Sprevak argue that computational structuralists cannot give an adequate account of computational individuation because, on a structuralist account, there are cases where it is indeterminate which (logical) functions are performed by which (physical) computational components. This is because the local structure of a physical system does not by itself fully constrain *which* specific mappings are permissible, beyond the basic structural morphism constraint.[4] So in cases where there is more than one possible mapping between a physical structure and a logical structure, computational structuralism cannot determine which computation is implemented. The solution advocated by both Sprevak and Shagrir is that a physical system must also possess semantic content in order to qualify as computational, and that this semantic content will fix the identity of otherwise indeterminate computational components. This solution is not available to the computational structuralists, whose aim is to individuate computational states and processes solely in terms of local physical structure.

To give a simple illustrative example; consider the hypothetical physical implementations of AND and OR described above. By mapping 1 to TRUE and 0 to FALSE, we got the result that Gate 1 implements AND, and Gate 2 implements OR. However, this assignment was essentially arbitrary. We could just as easily have done the opposite, mapping 1 to FALSE and 0 to TRUE. Since AND and OR are logical duals—i.e., their truth tables are structural inverts of one another—this mapping would flip the identity of the components: Gate 1 would implement OR and Gate 2 would implement AND (see Tables 7 and 8). Thus, at least according to computational structuralism, the computational identity of Gates 1 and 2 is indeterminate between AND and OR.

There are technically two ways that this alternative interpretation could be accomplished. First, the binary digits 0 and 1 could be remapped to TRUE and FALSE respectively, flipping the identity of the components (this is what we did above). Second, the mapping from binary digits to truth values could be kept the same, while 5V is instead indicated by 0 and 0V is instead indicated by 1, again resulting in the com-

[4] Note that this is a distinct problem from the triviality concern raised by Putnam (1988). Even if the computational structuralist can provide additional constraints on the mapping that avoid triviality, they will still be faced with the indeterminacy of the non-trivial mapping for every structurally dual pair.

Table 7 Gate 1 interpreted as OR

Input 1	Input 2	Output
F	F	F
F	T	T
T	F	T
T	T	T

Table 8 Gate 2 interpreted as AND

Input 1	Input 2	Output
F	F	F
F	T	F
T	F	F
T	T	T

putational identity of the physical components being flipped. The latter indeterminacy we see as simply coming down to the arbitrary nature of assigning Boolean values to physical states. It is the former sense of indeterminacy which is the more interesting kind, i.e., the fact that the computational identity of pairs of physical components implementing logical duals is structurally invertible.

As Shagrir (2001) has demonstrated, the problem also generalises to other cases, indeed to all pairs of structurally dual logical functions (such as IFF and XOR), and thus constitutes a general objection to computational structuralism—at least, provided that indeterminacy is seen as a problem. There has been much debate about whether computational structuralists might be able to live with this indeterminacy, and that discussion is ongoing (see e.g. Dewhurst, 2018a, Coelho Mollo, 2018, Shagrir, 2018, Lee, 2018, Miłkowski & Fresco, 2019, Fresco et al., 2021). Doherty (2021) has recently demonstrated that this indeterminacy is indeed a problem for computational structuralism, one that is more serious than has been previously recognised. In the next section we will present Doherty's argument that structural indeterminacy is symptomatic of an underlying indiscernibility problem, one that is analogous to an objection faced by mathematical structuralists.

2 From indeterminacy to indiscernibility

Doherty (2021) has shown that if computational structuralists are committed to functional indeterminacy with respect to structural duals (as discussed in the previous section), then they are committed to the indiscernibility, and hence identity, of the binary digits themselves, such that the binary digit 0 equals the binary digit 1. As this is presumably an absurd conclusion, we assume it is something computational structuralists must avoid. In this section we will present Doherty's argument, starting with an analogous objection to mathematical structuralism before applying the same objection to computational structuralism.

It should be noted that Doherty targets her objection at what she calls ‘pure’ computational structuralism, such as Chalmers (1996) and Dewhurst (2018a), but suggests that it could be extended even to ‘hybrid’ structuralists such as Piccinini (2015) and Coelho Mollo (2018), who invoke additional resources beyond causal structure—such as (proper) functions—to individuate computational states. In Sect. 4 we will argue that even pure structuralists can resist Doherty’s attempt to reduce them to absurdity by appealing to a Hilbertian conception of structure found in the philosophy of mathematics. We will therefore focus on pure structuralist accounts in what follows; if these can be defended against the indiscernibility objection, then hybrid accounts will also be safeguarded.

2.1 The indiscernibility objection to mathematical structuralism

In the philosophy of mathematics, non-eliminative structuralist accounts (such as ante rem structuralism) hold that mathematical structures and mathematical objects both exist and are the proper referents of our mathematical theories, but that the former have ontological priority over the latter—objects are ‘merely’ positions in a structure. The canonical accounts of mathematical structuralism are given by Hellman (1989), Resnik (1981), Parsons (2004), and Shapiro (1997)—only the latter two defend a *non-eliminative* structuralist account. The indiscernibility objection against non-eliminativist structuralism has recently been the focus of much attention, essentially arguing that these accounts are committed to the structural identity of patently non-identical mathematical objects. Versions of this objection can be found in the likes of Burgess (1999), Keränen (2001), Parsons (2004), MacBride (2006), Linnebo and Pettigrew (2014), and Wigglesworth (2018). We will briefly review the objection before applying an analogous version of it to computational structuralism.

The classical example is given by the imaginary complex numbers i and $-i$ (i.e., $a + bi$ and $a - bi$). These mathematical objects are shown to have the same structural properties by defining a (non-trivial) automorphism on the complex field; i.e., an isomorphism from the field of the complex numbers to itself which is not simply the identity mapping. Any structure over which we can define an automorphism is defined in the debate as *non-rigid*, and the objection can be formulated using any non-rigid structure. A simpler example is the additive integers, which are non-rigid because there is an automorphism which maps 1 to -1: the bijective function $\boldsymbol{f}$: $(\mathbb{Z}, +) \rightarrow (\mathbb{Z}, +)$ given by $\boldsymbol{f}(n) = -n$, $\forall n \in \mathbb{Z}$.[5] In each case it turns out that both putatively distinct objects ($i/-i$ and $1/-1$) have in fact all the same structural properties.

Proponents of the indiscernibility objection argue that the structural identity of these mathematical objects put a non-eliminativist structuralist between a rock and a hard place. Either they maintain that these objects are distinct in virtue of some non-structural (i.e., intrinsic) properties and so are more than ‘merely’ positions in a structure—violating the characteristic structuralist metaphysics—or, they are forced to accept embarrassing identity statements such as $1 = -1$, in virtue of the structural indiscernibility of these objects. MacBride concludes that non-eliminativist structural-

[5] Technically speaking this is true only when the integers are considered as an additive *group*, rather than a ring.

ism is either "old news or bad news" (MacBride, 2006, p. 64). There have been attempts to defend non-eliminativist structuralism against this objection (see e.g. Ladyman, 2005, Leitgeb & Ladyman, 2008, Shapiro, 2008), but so far none has been entirely successful. In Sect. 3 we will consider a novel solution drawing on the early structuralism of David Hilbert, but first we will review a recent application of the indiscernibility objection to the case of computational structuralism.

2.2 The indiscernibility objection to computational structuralism

Doherty (2021) demonstrates that a version of the indiscernibility objection to mathematical structuralism can also be applied to computational structuralism. She begins by establishing that if there is indeterminacy in the computational states then there is also indeterminacy in their logical status. Doherty defines a *computational state* in terms of digits, i.e., the Boolean values 0 and 1 (in the case of binary computation). As we clarified in the previous section, these should not be understood as abstract states (like the truth-values TRUE and FALSE), but rather as a way of referring to the computationally relevant properties of a physical system that implements a computation (again, for the computational structuralist these will be structural properties). If these computational states (digits) are fixed determinately then there is no way for the corresponding truth-values of the physical states to remain indeterminate. It might be arbitrary which truth-value we indicate with which digit, but we will at least be able to tell them apart.

More importantly, the logical determinacy of the digits entails that each physical state of the computer can be assigned a truth-value under the implementation (assuming we have already decided which physical state corresponds to which digit). Recall, for example, that to ensure that Gate 1 implemented AND and Gate 2 implemented OR we mapped 1 (indicating 5V) to TRUE and 0 (indicating 0V) to FALSE. The mapping from digits to truth-values will therefore also fully determine the identity of the truth-functions, since the (logical) function performed by a logical dual is determined by the assignment of truth-values to the digits it transforms. On the other hand, if there is indeterminacy in the assignment of the digits to the physical states (i.e., functional indeterminacy), then the truth-values will also be indeterminate.

This is all to say that the indeterminacy of the truth-values is symptomatic of a more fundamental indeterminacy in the computational digits.[6] Doherty thinks that the functional indeterminacy problem is more worrying than the logical indeterminacy problem, because if there is indeterminacy in the computational digits themselves (and not just their logical values), then computational structuralists are in fact committed to saying that the digits are *structurally indiscernible*. This means that they are also structurally *identical* because, at least for the *pure* computational structuralist, the identity of a computational digit is exhausted by the structure of a computational process, such

[6] Papayannopoulos et al. (forthcoming) draw a very similar distinction between 'interpretative' and 'functional' indeterminacy in computational systems, where functional indeterminacy relates to how *physical* states are grouped together and then mapped to *abstract* states, while interpretative indeterminacy relates to how these *abstract* states are then mapped to *logical* states. Our functional indeterminacy is the same as theirs, and our logical indeterminacy appears to be equivalent to their interpretative indeterminacy.

that digits with identical structural profiles are themselves identical.[7] This identity then leads to an apparent contradiction along the lines of (binary digit) 0 = (binary digit) 1, rendering the structuralist position absurd and ultimately unsustainable—or so Doherty argues. Pure computational structuralists are left in a similar position to the mathematical structuralists, as they are forced to accept that there are not really *two* binary digits, but rather just a single self-identical digit-type. Such a result would clearly render computation over binary digits impossible—surely a conclusion that the computational structuralist must avoid.

To resist this result the structuralist could find some supplementary extra-structural, non-semantic means of distinguishing the digits, as many modern structuralist accounts attempt to do by appealing to mechanistic, functional, or teleological resources (Piccinini, 2015, Coelho Mollo, 2018). However, Doherty doubts both whether there is any mechanistic or teleo-functional asymmetry in the behaviour of the computational digits, and also whether these kinds of resources (if they are available to the structuralist) can be made compatible with the requirement that computational explanation should be medium-independent. Of course, a proponent of the teleo-functional mechanistic approach might have the resources available to respond to these doubts (see e.g. Coelho Mollo, 2019), but for our purposes the important point is that all of these resources are going to be *extra-structural*, and therefore not available to the *pure* structuralist who wants to avoid appealing to anything other than structural features for their account of computation. As we have said previously, our focus here will be on providing a defence of pure structuralist accounts of physical computation, which should also suffice to defend hybrid accounts of the likes proposed by Piccinini and Coelho Mollo, even without appealing to any extra-structural resources such as teleological functions.

Doherty gives a formal proof of the indiscernibility of structurally individuated computational states that we will briefly rehearse here. For our purposes nothing much rests on the details on this proof, which simply demonstrates in formal terms the intuitive point that logical duals are structurally indiscernible (and hence, absent any other constraints, type-identical), but we hope that it might make clear the strength of the indiscernibility objection to computational structuralism. Doherty begins by defining a non-trivial automorphism on a simple system (akin to the automorphism used to formulate the objection in the mathematical case). An automorphism is simply a mapping which permutes the set while preserving the structure. Defining a (non-trivial) automorphism between the computational digits, then, amounts to a formal means of proving that they can be swapped while preserving the structure of the set, and so also proving that their structural profiles are identical. Doherty defines the automorphism as: $f : S \to S$ given by $f(x) = \neg x, x \in S$, for a set S of physical states. We can take S, for example, to be the set of the binary digits 0 and 1. Doherty then proves that f is not the identity function (so the automorphism is non-trivial) and

[7] Of course, much rests on how we define "identical structural profile", which Dewhurst (2018a) takes to imply the non-equivalence of putatively identical computational systems that differ only in terms of incidental features of their physical implementation. Miłkowski and Fresco (2019) have argued that this consequence is not inevitable, and we hope that structural equivalence can be defined so as to avoid it and thereby retain computational equivalence, but the threat from indiscernibility that we discuss here will remain even if this can be done.

that f is indeed an automorphism. Thus she proves f to be a function which formally establishes the identity of the binary digits in a simple computational system.[8]

In lay terms, this result obtains because the binary digits 0 and 1 have the same structural profiles, in the same sense that $i/-i$ or $1/-1$ have the same structural profiles. This can be seen more clearly by reflecting on the indeterminacy of the logical duals introduced in §1. The reason that the identity of the physical components implementing AND and OR is indeterminate is that they share the same formal structure, such that either component could play either role, depending on how the digits are interpreted. The digits are indeterminate in the same sense, and as a consequence they are also structurally *indiscernible*, such that if structural properties are all we have to go on they seem to be *identical*—which is nothing short of an absurd result!

It is worth saying a little more about how the indiscernibility objection is distinct from the indeterminacy objections discussed in §1. In Shagrir and Sprevak's objection there is no issue with *distinguishing* the functions. The issue, rather, is that there is nothing to decide which physical component should be cast in which computational role, since either of the *distinct* computational components could play the part of either *distinct* logical function. In Doherty's objection, structuralists cannot even maintain that the computational digits are distinct, and are thus committed an absurd identity claim, $0 = 1$ (or the identity of the binary digits). The objections are therefore different, but, as Doherty shows, they issue from the same root of the structural indiscernibility of the binary digits.

In the next section we will develop a response to this objection by drawing on Hilbert's earlier conception of mathematical structure. Before moving on, there are a couple of additional points we would like to note.

First, there are pure structuralist accounts of computation which ground computational identity in the identity of the physical states that the implementation maps to, such as Dewhurst (2018a). Doherty specifically addresses Dewhurst's proposal later in her 2021 paper. She concedes that it would avoid the indiscernibility objection by distinguishing any two computational digits implemented by distinct physical states, but points out that it does so at the price of failing to provide a medium independent account of computation, in which the same computational digits can be realised by distinct physical states across different mediums. Dewhurst was the first to point this out regarding his own proposal, and it has later been discussed by Coelho Mollo (2018) and Miłkowski and Fresco (2019), both of whom seek to defend a structuralist account of computation while avoiding this consequence. In doing so, however, they risk becoming vulnerable to Doherty's indiscernibility objection in a way that Dewhurst is not, as their commitment to medium-independence means that they cannot

[8] Doherty is concerned here with the binary case, which will therefore also be the focus of our discussion. Cases of indeterminacy or indiscernibility in possibly non-binary systems, such as that described by Shagrir (2018), will introduce additional complications. Whether (and how) we can define an automorphism in systems with the potential for more than two digit-types will depend on how we interpret their physical states. In Shagrir's original presentation of the (physical) tri-stable system, he offers two *binary* digital interpretations, to which Doherty's automorphism could be straightforwardly applied. A ternary digital interpretation of this system would require a three-valued logic, such as those defined by Lukasiewicz (1920/1970) or Kleene (1938). Both of these logics do include a NOT operator that could be used to generate an automorphism similar to the one Doherty uses in the binary case, although each logical system would require its own separate treatment that we will not pursue here.

appeal to the specific physical details of computational components in order to render them discernible (see (Doherty, 2021, sec. 8) for further discussion). In the next section we will aim to give a defence of pure computational structuralism which defuses this objection, without relying on the dead end of abandoning medium independence.

Second, we think it is independently interesting to see how the same kind of indiscernibility objection can be applied in two quite different areas of enquiry: abstract mathematical ontology and the philosophy of physical computation. This is of course not entirely surprising, as the objection targets structuralist accounts in both cases, and the latter area is concerned with the physical implementation of abstract computations, i.e., mathematical operations. However, these two areas have not recently interacted much, and we take it to be a very interesting result that despite the substantial differences between computational and mathematical structuralism, the indiscernibility objection plays out in the same way. What is more, in both cases the objection brings into doubt the common structuralist thesis: that the identity of an object is exhausted by its *structural* identity. It attempts to force structuralists—computational and mathematical alike—to appeal to extra-structural resources to identify and individuate the objects in their ontology. The indiscernibility objection is a powerful challenge to both kinds of structuralism, one that should be understood as threatening them with a potentially fatal *reductio ad absurdum*. However, the connection also offers a potential route to salvation for computational structuralism, by adapting a recent defence of mathematical structuralism *also* provided by Doherty (2019). In the next section we will introduce this defence, before finally applying it to computational structuralism in Sect. 4.

3 Hilbertian structuralism and indiscernibility

While she is the original source of the indiscernibility objection to *computational* structuralism, Doherty (2019) has in fact provided a defence of *mathematical* structuralism against the same indiscernibility objection by drawing on the early mathematical structuralism of David Hilbert (1862–1943). Our hope is that Doherty's very own defence can actually be adapted and made applicable to the computational case, in order to resolve her own objection to computational structuralism. There is a long answer and a short answer to the question of how Hilbert's early mathematical structuralism is able to avoid the indiscernibility objection. In this section we will set out both, beginning with the short answer in Sect. 3.1, elaborating on Hilbert's structuralism in Sect. 3.2, and finally presenting the long answer in Sect. 3.3. This will provide the necessary background context and theoretical machinery to apply an equivalent defence to *computational* structuralism in Sect. 4.

3.1 The heart of the defence

The short answer to how the Hilbertian conception of mathematical structure can avoid the indiscernibility objection is that it resists the move from the structural *indiscernibility* of mathematical objects to their *identity*. This is just to say that mathematical structuralists should reject the Principle of the Identity of Indispensables (PII), sometimes known as Leibniz's Law, i.e.,

> **PII.** Necessarily, for any x and any y, x is identical to y if for any P, x has P iff y has P. Formally, $\forall x \forall y(\forall P(Px \leftrightarrow Py) \rightarrow x = y)$.

The PII remains a contested metaphysical principle to this day, and rejecting it means that the mathematical structuralist can hold that there can be *non-identical* yet structurally indiscernible mathematical objects, like the two complex roots of -1. That is to say, structuralists can claim that there are mathematical objects with *identical* structural profiles but which are nevertheless *distinct*, in the same way that Max Black maintains it is possible for two qualitatively identical spheres in a completely symmetrical universe to be numerically distinct despite being indiscernible (Black, 1952). The structuralist can therefore accept that the automorphism proof establishes that the *structural profile* of the permuted objects is identical, without conceding that this leads to their actual numerical identity, allowing them to avoid the apparent absurdity generated by the indiscernibility objection.

This rejection of PII provides an important insight into the argumentative structure of the indiscernibility objection.[9] However, rejecting PII can only constitute a partial solution to defending structuralism against the indiscernibility objection. PII might be contestable, and rejecting it might be the key to avoiding the objection, but structuralists need to provide a principled reason to reject it beyond the mere fact that it would be convenient for them—one that is grounded in, and compatible with, a structuralist ontology. Shapiro, for example, whose own ante-rem structuralism inherits much from Hilbert, rejects PII without providing any justification, and his solution is accused of being *ad hoc* (see Shapiro, 2008, p. 287; MacBride, 2006; Keränen, 2001). However, in Hilbert's original early conception of mathematical structuralism it is possible to find the kind of non-*ad hoc* reason to reject PII that is required to buttress a robust defence against the indiscernibility objection.[10]

In order to give the longer answer as to how Hilbert's early structuralism avoids the indiscernibility objection, we must explain his principled reasons for rejecting PII. This will first involve detailing the relevant aspects of Hilbert's structuralism and, in particular, his conception of the relationship between theory and ontology in mathematics.

[9] We take Hilbert's rejection of PII to be more powerful and comprehensive than the other defences of structuralism offered in the literature, such as Ladyman's appeal to weak discernibility relations, i.e. relations that the structural duals stand in to each other but do not stand in to themselves, like *is the additive inverse of* for i and $-i$ (Ladyman, 2005). Even Ladyman admits that such relations cannot distinguish all structurally indiscernible mathematical objects (Leitgeb & Ladyman, 2008).

[10] Note that Hilbert didn't *directly* propose a solution to any objection from structurally indiscernible non-identical objects. The first of these objections was raised at least 56 years too late for Hilbert to have even read it. However, as Doherty (2019) has argued, his early structuralist account—predating his more infamous formalism—already provides us with a solution to the problem, because rejecting PII follows as an immediate consequence of a central thesis Hilbert *did* explicitly endorse, namely Hilbert's Principle.

3.2 Hilbertian structuralism and Hilbert's principle

The clearest formulation of Hilbert's structuralist thesis can be found in his defence of the *Grundlagen der Geometrie* (1899b) against Frege's critique.[11] Hilbert writes to Frege,

> ... it is surely obvious that every theory is only a scaffolding or schema of concepts together with their necessary relations to one another, and that the basic elements can be thought of in any way one likes (Hilbert, 1899a, p. 40).

Hilbert's re-axiomatisation of classical geometry is constructed so that it contains only six non-logical 'basic elements', i.e. the geometric primitives: $point$, $line$, $plane$, $congruence$, $lies\ on$, and $between$ (Hilbert, 1899b, §1). For Hilbert, what matters is the structure defined by the logical relations, not the objects/relations referred to by the theory's primitive terms. This contrasts starkly with Frege's conception of a theory as a deductively organised set of true thoughts, referring to a fixed set of mathematical objects (Frege, 1899, p. 36, 1903, p. 281). Frege goes so far as to say that Hilbert's primitive indeterminacy renders *Grundlagen der Geometrie* "a failure" (Frege, 1900, p. 90). However, this very same feature is the key to Hilbert's groundbreaking proofs of the consistency and independence of the axioms of classical geometry (Hilbert, 1899b).

This feature of Hilbert's conception of a mathematical theory is important to set out, i.e., that Hilbert takes his primitives to be indeterminate in the sense that they are inherently reinterpretable. The reinterpretability of the primitives makes it possible to assign them meaning using other mathematical theories in which we can prove the truth of the axioms. Proving the truth of the axioms establishes the consistency of the geometric theory in virtue of the structure common to both the geometric theory and its reinterpretation.

In particular, Hilbert reinterprets his geometric axioms using the domain Ω: a fragment of the real numbers specified by Hilbert in (Hilbert, 1899b, §9). For example, he reinterprets his second axiom of connection:

> I, 2. For every two points there exists at most one line which lies between those points

as follows:

> [I, 2.$^{\Omega}$] For any distinct pair of pairs of real numbers $\langle\langle a, b\rangle, \langle c, d\rangle\rangle$ there is a unique ratio $[e : f : g]$, such that both $ae + bf + g = 0$ and $ce + df + g = 0$

Where "distinct pair" entails that $\langle a, b\rangle \neq \langle c, d\rangle$, since there are infinitely many lines through a single point. The remaining axioms are likewise reinterpreted. Hilbert's reinterpretation makes it straightforward to prove the truth of the reinterpreted axioms using the reals, and this proof establishes the consistency of Euclidean geometry *relative to the real number field* by model-theoretic reasoning. Thus, Hilbert conceives

[11] The fact that Hilbert advocated a Dedekind-inspired variety of structuralism in the early 1900's before his invention of proof-theory is perhaps little known, but is not controversial. See Hilbert (1899a, pp. 40–41, 1900, pp. 50–51). See also Resnik (1981, p. 202), Sieg (2008, p. 467), Sieg (2014, pp. 135–138), Lindström and Palmgren (2008, p. 17), Parsons (1990, p. 336, 2004, p. 71) and Shapiro (2005, p. 62).

of a theory as a 'conceptual scaffold', in which the primitives are constrained only by the requirement of satisfying the structure defined by the axioms of the theory. This approach is described by Hilbert as "a tremendous advantage" (Hilbert, 1899a, p. 41), vindicated by his success in proving the consistency and independence of the axioms of classical geometry.

If what matters to Hilbert is fixing determinate reference to a structure, rather than to the objects of a theory, then a question arises as to how Hilbert ensures that the theory successfully refers to a structure. This brings us to the second important feature of Hilbert's conception of a mathematical theory: according to Hilbert, all that a theory requires to ensure that it latches on to a structure is internal consistency. This is often known as Hilbert's Principle[12], the most famous formulation of which is:

> if arbitrarily chosen axioms together with everything which follows from them do not contradict one another, then they are true, and the things defined by the axioms exist. For me that is the criterion of truth and existence. (Hilbert, 1899a, pp. 39–40)

This principle requires some careful unpacking. Understood as a general ontological principle it is of course absurd. However, in context it is clear that Hilbert restricts the principle to the domain of mathematics where, as we have just seen, Hilbert's primary concern is with fixing reference to *structures* (1899, p. 36). It should also be noted that Hilbert's Principle can be read as indicating that Hilbert's eliminative approach to mathematical structures supports a more general eliminativism with regard to mathematical objects.[13] In what follows we will treat Hilbert as non-eliminative about both mathematical objects and structures, following Doherty (2019). However, we want to make clear that nothing much hangs on this question of whether Hilbert should be understood as an eliminativist. What variety of structuralism one wants to advocate or attribute to Hilbert is independent of whether Hilbert's approach can be shown to provide a solution to problems faced by structuralists of different ilks, and in this case the relevant objection besets *non-eliminative* structuralists. Regardless of whether Hilbert's Principle is read as ontologically inflationary or ontologically deflationary, it amounts to an alignment of consistency and existence for mathematical structures, such that all that is needed to ensure the existence of a structure is for it to be defined by a consistent set of axioms.[14]

The indiscernibility objection to mathematical structuralism concerns structurally identical *objects*. Given what we have said, what can Hilbert's structuralist account tell us about objects at all? For non-eliminative structuralists, mathematical objects exist but are ontologically dependent on structures, in the sense that they are reducible to a position in a structure. Therefore, establishing the existence of a structure is tantamount to establishing the existence of its positions. So, on the Hilbertian view, the consistency of the axiom set establishes the existence of the structure it defines, and

[12] Ferreirós (2009) and Pudlák (2013, p. 602).

[13] See e.g. Resnik (2018, p. 1).

[14] It is worth noting that modern structuralist accounts tend to advocate something akin to this proposal, but with a higher bar than mere consistency. Shapiro, for example, grounds the existence of a structure in the *coherence* of the axioms (Shapiro, 2005, pp. 69–71).

reifications of the positions in the structure are properly thought of as mathematical objects.

However, the thesis that a consistent axiomatisation can guarantee the existence of mathematical objects, in any sense, seems to be more controversial than a thesis that merely guarantees the existence of a mathematical structure. Let us say a few things to quell any uneasiness here. The first thing to emphasise is that this thesis is only needed because we are providing a defence of *non-eliminative* structuralists who, of course, already endorse it. Looking ahead, we don't intend to defend or rely on this thesis when we give a Hilbertian approach to computational individuation. We only need to transpose the uncontroversial thesis that consistently/coherently defined *structures* exist. Furthermore, it should be noted that the thesis comes along with a certain conception of mathematical objects such that they are, contra Platonism, only 'thin' idealised (or theoretical) objects. Hilbert is always very clear on this point, for example in his *Grundlagen der Mathematik* he speaks of a theory as "completely detached from concrete reality" such that it "...has nothing more to do with real objects or with the intuitive content of knowledge. It is a pure thought construction..." (Hilbert, Hilbert 1922, p. 3).[15]

In summary, Hilbert's Principle is that the consistency of a theory's axioms entails the existence of the structure they define, and thereby the existence of the positions in the structure, which can be understood as the objects of that theory. In the next section we will use this principle to develop a Hilbertian response to the indiscernibility objection against mathematical structuralism.

3.3 The Hilbertian solution to the indiscernibility objection

Finally, we can now give the full presentation of the Hilbertian solution to the indiscernibility objection. Recall that the objection is that structuralists are forced to treat as identical intuitively distinct mathematical objects that have the same structural profile (i.e. are structurally indiscernible), such as the complex roots of -1.

We saw that Hilbert's short answer to this problem was to reject PII. We can now say that Hilbert's longer answer is that if the axioms of a consistent axiomatisation specify a structure with distinct but indiscernible positions, then distinct but indiscernible mathematical objects must unproblematically exist. As we saw, this result follows as a consequence of Hilbert's Principle, which entails that whatever is defined by a consistent set axioms must exist. What the axioms define are structures, and objects are merely the structure's positions such that, if a structure exists so do its positions, and if a position exists, then so does a mathematical object. Therefore, if there is even one consistent axiomitisation which defines a structure with at least two indiscernible positions, then it will be the case that there exists, without absurdity, distinct but structurally indiscernible mathematical objects. This is the principled justification for

[15] Of course, Hilbert doesn't take mathematical theory to be irrelevant to real objects or knowledge about the world, but again takes a structuralist approach. He continues on in the same passage to say that: "Nevertheless, this framework has a meaning for knowledge of reality, in the sense that it presents *a possible form of actual connections*." (Hilbert, 1922, p. 3, emphasis added).

Hilbert's rejection of PII, going right to the heart of his conception of a mathematical theory.

Due to its importance, let us unpack this principled justification a little more. For an illustrative analogy of Hilbert's conception of a theory, consider an isosceles triangle (defined by a consistent set of axioms) constructed by projecting the lines and angles of intersection of the right-hand side of the triangle to the left-hand side. We can define a rotation of 180° around its axis of symmetry, mapping the right-hand side of the triangle to the left-hand side. The right-hand side and the left-hand side of the triangle have the same structural profiles in the sense that they are congruent. However, this is not in tension with the fact that there are two distinct sides composing the triangle, specified axiomatically. In such a case there is no absurdity in the claim that the symmetric sets of points and lines composing each side of the triangle are made up of distinct but structurally indiscernible geometric objects. Drawing back from the analogy, we can say that all that the existence of a non-trivial automorphism establishes is that there is a symmetry in the relevant set. On Hilbert's conception of a theory we require a further reason to collapse this symmetry into an identity.[16]

Hilbert himself even discusses a relevant example:

> ...The proposition 'There are two square roots of -1' is true, and *the existence of two such roots is proven*, as soon as the axiom 'There are *two* roots of -1' can be added to the other arithmetical axioms, without raising the possibility of contradiction, no matter what conclusions are drawn. (Hilbert, Hilbert 1899a, pp. 39–40, emphasis added)

Of course, it was Hilbert who was to establish—by means of *this* conception of a theory—the model-theoretic (and later, proof-theoretic) consistency of the complex numbers. Thus the (relative) consistency of complex analysis acts not as a counterexample to structuralism but, if anything, as a counterexample to PII for the case of structurally individuated mathematical objects.[17]

To summarise, Hilbert is able to avoid the indiscernibility objection because his conception of a theory justifies his rejection of PII. In particular, Hilbert can reject PII on account of his conception of the primitives of a mathematical theory as inherently reinterpretable, such that the only thing a consistent theory determinately refers to is a structure, and the only objects it can be said to determinately refer to are the positions in that structure (thought of as a 'thin', idealised kind of object). Hilbert thereby turns the tables on proponents of the indiscernibility objection, because on this view it is no absurdity to point out that there is a symmetry in the structure as shown by the presence of an automorphism. Furthermore, if it *was* a consequence of the axiom set that $i = -i$, this would give a very immediate contradiction with the axioms of complex analysis, demonstrating that the higher-order structure and associated ideal objects of any theory containing such a theorem do not in fact exist.

[16] Note that were we to *identify* the elements of a non-trivial automorphism with each other, this would demote the function to a trivial automorphism.

[17] Of course, the swathes of indiscernibility objections against mathematical structuralism—e.g. the complex roots, additive integers, cyclical rings, and even simple unlabelled graphs in graph theory—all serve to provide further examples where this approach could be applied.

4 Applying the Hilbertian solution to the computational case

We come now to our final task of applying Hilbert's conception of a theory to computational structuralism, in order to provide a defence against the novel indiscernibility objection raised by Doherty. Again, our sights will be set on defending pure structuralists like Dewhurst (2018a), since it is they who are most vulnerable to the objection. Hybrid structuralists like Piccinini (2015) and Coelho Mollo (2018) are able to draw on other resources, such as proper functions, but the Hilbertian solution will also ensure that they can safeguard the medium independence of physical computation.

We should emphasise at this point that it doesn't matter that the two indiscernibility objections presented in Sect. 2 target very different kinds of 'structuralist' accounts: the broad church of mechanistic/causal accounts of computation on the one hand and non-eliminative structuralism about mathematical objects on the other. Our intention here is not to transfer the specific technical details of Hilbert's re-axiomatisation of classical geometry to the case of physical computation, but rather to draw an analogy between the solution he proposes for the case of indiscernibility in pure mathematics, and a potential solution to the case of indiscernibility in physical computation.[18] As we aim to show in this section, the details of Hilbert's solution—and to some extent even his wider conception of a theory—are general enough to facilitate its adaption to the computational case.

Our demonstration that Hilbert's structuralism has the potential to absolve pure computational structuralists from the charge of absurdity falls into three parts. First, we will show that if pure computational structuralists adopt the central insight of Hilbert's conception, namely, the rejection of PII, then they will avoid the indiscernibility objection. Next, we will offer some reflections as to whether different accounts of computational structuralism can reject PII without being *ad hoc*. Finally, we will consider some further implications of this approach, cashed out in terms of the practical implications of our Hilbertian solution to the indiscernibility objection.

4.1 The Hilbertian solution applied to computational structuralism

In the case of mathematical structuralism, Doherty (2019) advocates a Hilbertian solution to the indiscernibility objection that amounts to rejecting PII and thereby accepting that there can be structurally indiscernible but numerically distinct mathematical objects (such as i and $-i$). For pure computational structuralists, the rejection of PII amounts to endorsing a particular view of the non-semantic individuation of basic components in a physical computing system. On this view, there can be two *structurally* indiscernible computational components which are nonetheless non-identical, such that the original problem of indeterminacy is also undercut. In this way, our response to the indiscernibility objections raised by Doherty (2021) can also be seen as a solution to the original indeterminacy objection raised by Shagrir (2001) and Spre-

[18] It is possible that the technical details of Hilbert's structuralism might also be applicable to the case of physical computation, and demonstrating this applicability is a project that we would be interested to pursue in future work, but our present aim is more modest: simply to demonstrate that a solution which is Hilbertian in spirit can be applied to the case of computational structuralism.

vak (2010). Put into Hilbertian terms, our theory of physical computation requires that there are two positions in the abstract computational structure (corresponding to the truth functions AND and OR) that must be filled by physical components, but it does not matter which component fills which position, or that they are structurally invertible.[19] We will now discuss this solution in a little more detail.

In the mathematical case, the Hilbertian solution meant accepting that there can be objects (like i and $-i$) that are structurally indiscernible but axiomatically distinct, in so far as a consistent theory implies that there are *two* such indiscernible spaces in the structure which must be filled by *distinct* objects. The situation is analogous in the computational case, where we can have consistent models that imply the existence of structurally indiscernible digits (0/1) and structurally invertible functions (logical duals like AND and OR), which are nonetheless both distinct. The physical implementations of these digits and functions will therefore also be structurally indiscernible/invertible, while nonetheless requiring two distinct objects in each case. The crucial point to realise is that it is their distinctness, not their discernibility, that is important for avoiding both the indiscernibility and indeterminacy objections. Just as it doesn't matter to Hilbert which (mathematical) object is i and which is $-i$, but only that there *are* two such objects, it also shouldn't matter to the computational structuralist which (physically implemented) gate is AND and which is OR, only that there *are* two such gates - and nor should it matter which (physically implemented) digit is 0 and which is 1, but only that there *are* two such digits to form the basis of binary computation. The computational structuralist should thus embrace both *both* indeterminacy and indiscernibility, while insisting that the distinctness of the components/digits is sufficient for a theory of computational implementation.

From this point of view, Doherty's objection can actually be seen as an advantage for the computational structuralist, since it diagnoses the root cause of the indeterminacy of logical duals as being found in their structural indiscernibility. Thus, by following Hilbert in responding to the indiscernibility objection by rejecting PII, the computational structuralists can also deflate the original indeterminacy objection. Pure computational structuralists are therefore justified in biting the bullet regarding precisely the kind of harmless indeterminacy that is to be expected on a *structuralist* account of computation. Dewhurst (2018a) presents a similar solution that Miłkowski and Fresco (2019) have recently argued bites one bullet too many, by implying that the *equivalence* of computational components might be impossible. The bullet that we advocate biting here is a distinct one, requiring that we accept there is no non-arbitrary way to assign truth values to logical duals, but we believe that logical indeterminacy of this kind is harmless, as physical components will continue to carry out the same computational role regardless of the truth value we assign to them. Our approach has no specific implications for computational equivalence, and is thus able to avoid biting the original bullet bitten by Dewhurst (2018a). It turns out that both indeterminacy and indiscernibility are harmless consequences of pure computational structuralism, just as Hilbert's original structuralist conception of mathematical objects implied a harmless kind of indiscernibility for some mathematical objects.

[19] The same of course goes for other logical duals, such as IFF and XOR.

4.2 Rejecting the PII for computational structuralism

So far things are looking up for pure computational structuralism. However, as was the case for mathematical structuralists, the computational structuralist should also provide a principled reason for rejecting PII, lest they be vulnerable to the charge that their solution is merely *ad hoc*. Computational structuralism is a broad camp, including not only mechanistic accounts but also causal and syntactic accounts more generally, and each will have to provide their own particular justification for rejecting PII, compatible with their own theoretical commitments. Detailing what each of these might look like would take us outwith the scope of this paper, but in what follows we provide some suggestions and reasons for optimism that such justifications can be provided.

The first thing we want to note is that there is nothing *incompatible* between the core commitments of computational structuralism and the rejection of PII. To reject PII is simply to take a stance on whether structurally indiscernible objects are necessarily identical or not. The computational structuralist cannot simply stop there, as they should provide some *internal* justification for rejecting PII, but there is no reason to think that this won't be possible. In order to maintain a pure structuralist attitude, this justification should ideally also be *structural* in nature, i.e., it should not have to appeal to any external, non-structuralist metaphysical principles. Structuralists could of course engage with the details of the ongoing metaphysical debates around PII, and provide a more general argument for rejecting it. However, we believe that they can get away with much less, by simply providing a principled reason to reject PII that is consistent with, and internal to, their structuralist account of computational implementation.

Once again we think that the broader Hilbertian conception of a theory can provide some promising resources for a computational structuralist to justify the rejection of PII. The essence of our suggestion is to adopt a Hilbertian conception of the relationship between an abstract computational model and its physical implementation. In the mathematical case, we saw that any mathematical structure that satisfies (is consistent with) the axioms of the theory can be said by Hilbert to exist.[20] Analogously, in the computational case we can simply say that any (physical) structure (or system) that satisfies (is consistent with) the abstract model is an implementation of that model. Of course, this will result in a very promiscuous account of computational implementation, but here we are interested primarily in the question of *individuation*, not triviality, and so we can allow that some other means of restricting the range of admissible implementations might be required. One possibility here would be to appeal to the kind of information-theoretic "simplicity criterion" proposed by Millhouse (2019), which could allow us to measure in objective terms the relative complexity of the relationship between specific physical and mathematical structures. While this might still result in a form of unlimited pancomputationalism, it would at least allow us to rank (in non-arbitrary terms) which computational models are more or less interesting for each candidate physical implementation. This kind of constraint would be consistent

[20] And consequently, that any object/reified position of that structure can also be said to exist. Note that at the time Hilbert was writing he aligned the concept of consistency with that of satisfiability, since model-theoretic proofs (which he invented) were the only known means of establishing consistency.

with the structuralist account on offer here, as it doesn't appeal to intrinsic properties of computational objects, but rather just to the formal relationship between physical implementation and computational model. The Hilbertian solution to indiscernability is also available to hybrid structuralists, who can appeal to additional non-structural resources such as proper functions in order to block pancomputationalism. These resources are not available to the pure structuralist, who might therefore have a harder time dealing with pancomputationalism, even if they are able to avoid indiscernability. Some pure structuralists might even be willing to accept pancomputationalism, but this would certainly be a controversial move (see e.g. Schweizer, 2019 for an example of this kind of approach). In any case, we do not want to commit either the pure or hybrid structuralist to any particular way of dealing with triviality here, and will just note that it is a further issue they may need to address once the current problem of indiscernibility has been resolved.

According to the Hilbertian structuralist account, then, Gate 1 can be said to implement *either* AND or OR, but not *both at once*, as its structure is consistent with either of these models, and vice versa for Gate 2. All that matters is that there is a pair of physical components structured such that they can fill either position in the computational model, and interact with each other such that they satisfy the structure of this model (i.e. transform digits in the correct way). This approach serves to justify rejecting PII in the following way – we can adapt Hilbert's Principle to create an equivalent principle for computational structuralists, essentially a restatement of simple mapping accounts of physical computation:

> **Computational Structuralist Principle** The existence of a physical computational structure is established when there is a homomorphism between an abstract computational structure and a physical structure, such that the physical structure can be interpreted as an implementation of the abstract computational structure.

Hilbert's justification for rejecting PII came from the fact that Hilbert's Principle entailed the falsity of PII. Likewise, here the Computational Structuralist Principle entails the falsity of PII for physical computation. For, if there is an abstract computational model that defines a system with a structural symmetry—i.e., two or more digits with indiscernible structural profiles—and can be mapped to a physical structure, then there exists a physical computational structure with structurally indiscernible but numerically distinct component-types, contra PII. We can take any example here, such as the implementations of the digits of the AND-gates and OR-gates described in Sect. 1. This constitutes a counterexample to PII for physical computation, and hence offers computational structuralists a principled reason to reject PII, paralleling Hilbert's own rejection of PII in the case of mathematical structuralism.

In this way computational structuralists can also turn the indiscernibility objection on its head; if there *was* a physical structure with only a single, self-identical component type and a single, self-identical digit type, then this structure would not count as an implementation of any abstract (binary) computational structure, and so could not be classified as a physical computational structure in the first place.[21] The computational structuralist has a principled internal reason for rejecting PII, and while this

[21] Note that a NOT-gate requires two distinct digit types, and thus still qualifies as a binary computational system, although as (Fresco et al., 2021, sec. 2.2) note, such a system by itself is able to avoid indeterminacy

justification might not be shared by their opponents, it at least saves them from the threat of contradiction via the absurd conclusion that the digit type "1" equals the digit type "0".

4.3 Forward- and reverse-engineering indeterminate components

The final thing we want to discuss are some practical implications of using Hilbert's conception of a theory to avoid absurdity and embrace the indeterminacy of computational systems with localised symmetries. For some, in particular the semanticists who originally raised the indeterminacy objection, this approach might seem unsatisfactory. After all, it is still the case that pure computational structuralists cannot determine which structural dual is which for any non-rigid computational system.

We want to question why such indeterminacy is still seen as a problem, rather than simply a harmless (though perhaps initially counter-intuitive) consequence of the structuralist conception of physical computation. To examine this issue further we will imagine two hypothetical characters, corresponding to two different kinds of theoretical project. The ***forward engineer*** is interested in computational implementation because she wants to build a physical system that implements an abstract computational model. The ***reverse engineer***, on the other hand, is interested in computational implementation because she wants to determine which abstract computations a particular physical system implements. The former corresponds to the kind of challenge faced by those involved in the engineering project of creating a functioning physical computer, while the latter corresponds roughly to the challenge faced by computational neuroscientists, who are tasked with determining what computations the brain might be performing, but could also include regular computer scientists trying to, e.g., reverse engineer a rival's technology.

For forward engineers, one might think that establishing the determinate computational identity of the components would be important, because they want the physical system to carry out specific computational operations. On closer inspection, forward engineers should not be concerned by our Hilbertian indeterminacy, as they can simply *stipulate* which component fulfils the role of each logical function, provided that this remains consistent with how they individuate the other components of the system they are designing. More specifically, this stipulation must be consistent with their assignment of truth values to whichever physical component plays the role of each digit in their system, such that the physical inputs and outputs of each gate match their corresponding function in the logical model. We saw already that any non-rigid computational structure could be rendered fully determinate if the computational digits were assigned a truth-value, and that this would in turn determine the computational identity of gates implementing structural duals. For forward engineers, the arbitrary nature of how we interpret logical duals should feel quite natural—these are after all systems *we* have designed. The Hilbertian solution is a comfortable one here since

Footnote 21 continued
concerns. Unary digital computation would of course only include a single, self-identical digit type, but we are concerned here only with binary digital computation, and unary computation is anyway an edge-case that is likely to appear somewhat strange under any account (see Maley, 2020 for some recent discussion).

there is no inherent problem in the fact that the computational system doesn't internally determine truth functions across its structural symmetries and, furthermore, the truth functions can be easily specified if one's engineering goal requires it.

Indeterminacy may initially seem more problematic for the reverse engineers, who cannot simply stipulate the identity of each indiscernible gate in the system they are studying—they are, after all, trying to *discover* the identity of its components, not merely label them. This problem, however, dissolves under the Hilbertian conception of a computational structure. Crucially, if we are not to beg the question against the computational structuralist, we must acknowledge that their view entails that all that matters for the identity of a (physical) computational component is that its structure corresponds with the structure of an (abstract) computational model. In the case of logical duals, this correspondence can be achieved without determining which component is which, provided that two such components can be identified *and distinguished from one another*. For example, Gates 1 and 2 fulfil this criteria for the logical duals AND/OR, which should be sufficient for computational individuation. Any further labelling or interpretation (of one as AND and the other as OR) is strictly not part of the core process of computational individuation, even if it might be pragmatically convenient or useful for some further process (such as the development of a theory in computational neuroscience). In this case a determinate assignment can also be arbitrarily specified—just the same as with the forward engineer—*since there is no fact of the matter for the reverse engineer to discover*. In other words, for computational structuralists engaged in reverse engineering there is nothing to discover beyond the inverse homomorphism (technically speaking, the coding) between an abstract computational structure (including any symmetries) and the physical structure that is thought to implement a computation.[22]

Thus, our Hilbertian solution should satisfy both the forward and reverse engineers, even though it concedes that certain computational components are both structurally indeterminate and indiscernible. Such components must nonetheless be numerically distinct in order to satisfy the requirements of the abstract computational model, even if there is no fact of the matter about the 'correct' assignment of truth-functions to either component. There are simply two equally viable candidates for two computational roles, but each role can only be fulfilled by one candidate at a time, and both roles must be filled. Fixing the identity of one component as AND, for example, determines that the other component *must* play the role of OR. Given this, it can no longer be considered an *objection* to computational structuralists that logical duals are indeterminate or indiscernible. It is instead simply an explicit *consequence* of their view, according to which the structural profile alone is all that can fix the computational identity of a physical component. This, we think, should satisfy any theoretical requirements of both our forward and reverse engineers.

[22] Codings are inverse homomorphisms which are injective (one-to-one) and as such can be 'decoded' as the reverse engineer requires.

5 Conclusion

Our central result is to have shown that it is possible to rescue even pure computational structuralism from the indiscernibility objection recently presented by Doherty (2021). It follows from this that hybrid accounts of computational structuralism can also escape this objection. To do so we recommend that computational structuralists of both kinds should reject the Principle of the Identity of Indiscernibles, but we recognise that they require some internal justification for doing so. We suggested that such a justification can be found by adopting and adapting certain key aspects of Hilbert's conception of the relationship between a theory and its subject matter. Computational individuation should be grounded not only in the structural profile of components, but also in other aspects of the structure, in particular the fact that the physical structure is an implementation of an abstract computational model with localised structural symmetries.

Another outcome of our argument is to demonstrate that Hilbert's early structuralism is a fruitful resource for computational structuralists, although it is clear that Hilbert's conception of mathematical theories will have to be carefully adapted if it is to be made use of in this way. Finally, we also take it to be very interesting that the *very same* indiscernibility objections that arise for mathematical structuralists can also be applied to computational structuralism, and that a similar solution to the first objection can be adapted for the second. This suggests that there might be some deeper connections between the two kinds of structuralist approach that deserve further study in future projects.

Our proposal for a wholesale adoption of an adapted Hilbertian approach to computational structuralism may seem radical, but we take it to be well-motivated given the power and influence of his approach to mathematical structuralism. It is easy to forget that it was Hilbert's conception of reinterpretable partially underdetermined theoretical primitives which gave birth to his now standard consistency and independence proofs, paved the way for the theory of general relativity, and not least, is the standard approach among working mathematicians and computer scientists to this day.

Acknowledgements We would like to thank the organisers and audience of the *Indeterminacy and Underdetermination* workshop held at University College Dublin in January 2020, the editors of this topical collection, and especially the two anonymous reviewers for this journal, whose comments helped us improve the paper significantly.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Black, M. (1952). The identity of indiscernibles. *Mind, 61*, 153–64.

Burgess, J. (1999). Review of Shapiro (1997). *Notre Dame Journal of Formal Logic, 40*, 283–91.

Chalmers, D. J. (1996). Does a rock implement every finite-state automaton. *Synthese, 108*, 309–333.

Coelho Mollo, D. (2018). Functional individuation, mechanistic implementation: the proper way of seeing the mechanistic view of concrete computation. *Synthese, 195*, 3477–3497. https://doi.org/10.1007/s11229-017-1380-5

Coelho Mollo, D. (2019). Are there teleological functions to compute? *Philosophy of Science, 86*(3), 431–452. https://doi.org/10.1086/703554

Copeland, B. J. (1996). What is computation? *Synthese, 108*, 335–59.

Dewhurst, J. (2018). Individuation without representation. *The British Journal for the Philosophy of Science, 69*(1), 103–16. https://doi.org/10.1093/bjps/axw018

Dewhurst, J. (2018). Computing mechanisms without proper functions. *Minds and Machines, 28*, 569–88. https://doi.org/10.1007/s11023-018-9474-5

Doherty, F. T. (2019). Hilbertian structuralism in the Frege-Hilbert controversy. *Philosophia Mathematica, 3*(27), 335–361. https://doi.org/10.1093/philmat/nkz016

Doherty, F. T. (2021). Computational Indeterminacy. *PhilPapers*. https://philpapers.org/rec/DOHCI-3.

Egan, F. (1992). Individualism, computation, and perceptual content. *Mind, 101*, 443–459.

Ferreirós, J. (2009). Hilbert, logicism, and mathematical existence. *Synthese, 170*(1), 33–70.

Fodor, J. A. (1998). *Concepts*. Blackwell.

Frege. G. (1899). Frege to Hilbert. In: G. Gabriel, H. Hermes, F. Kambartel, C. Thiel, A. Veraart, B. McGuinness, & H. Kaal (Eds.), *Gottlob Frege: Philosophical and mathematical correspondence* (pp. 34–38). Blackwell

Frege, G. (1900). Frege to Hilbert. In: G. Gabriel, H. Hermes, F. Kambartel, C. Thiel, A. Veraart, B. McGuinness, & H. Kaal (Eds.), *Gottlob Frege: Philosophical and mathematical correspondence* (pp. 49–50). Blackwell.

Frege, G. (1903). On the foundations of geometry: First series. In E. H. W. Kluge (Ed.), *On the foundations of geometry and formal theories of arithmetic* (pp. 22–37). Yale University Press.

Fresco, N. (2015). Mechanistic computational individuation. *Erkenntnis, 80*, 1031–53.

Fresco, N., Copeland, J. B. & Wolf, M. J. (2021). The indeterminacy of computation. *Synthese*

Hellman, G. (1989). *Mathematics without numbers: Towards a modal-structural interpretation*. Oxford University Press.

Hilbert, D. (1899a). Hilbert to Frege . In: G. Gabriel, H. Hermes, F. Kambartel, C. Thiel, A. Veraart, B. McGuinness, & H. Kaal (Eds.), *Gottlob Frege: Philosophical and mathematical correspondence* (pp. 38–43). Blackwell

Hilbert, D. (1899b). *Grundlagen der Geometrie*. Teubner. English translation of 10th edition by L. Unger, Chicago: Open Court.

Hilbert, D. (1900). Hilbert to Frege. In: G. Gabriel, H. Hermes, F. Kambartel, C. Thiel, A. Veraart, B. McGuinness, & H. Kaal (Eds.), *Gottlob Frege: Philosophical and mathematical correspondence* (pp. 50–51). Blackwell

Hilbert, D. (1922). Grundlagen der Mathematik, lecture notes by Bernays. In W. B. Ewald, M. Hallett, M. Ulrich, & W. Sieg (Eds.), *David Hilbert's lectures on the foundations of arithmetic and logic 1917–1933* (pp. 431–527). Springer.

Keränen, J. (2001). The identity problem for realist structuralism. *Philosophia Mathematica, 3*, 308–30.

Kleene, S. C. (1938). On notation for ordinal numbers. *Journal of Symbolic Logic, 3*, 150–155.

Ladyman, J. (2005). Mathematical structuralism and the identity of indiscernibles. *Analysis, 65*(3), 218–21.

Lee, J. (2018). Mechanisms, wide functions, and content: Towards a Computational Pluralism. *British Journal for the Philosophy of Science*. https://doi.org/10.1093/bjps/axy061

Leitgeb, H., & Ladyman, J. (2008). Criteria of identity and structuralist ontology. *Philosophia Mathematica, 16*(3), 388–396.

Lindström, S., & Palmgren, E. (2008). Introduction: The three foundational programmes. In: S. Lindström, E. Palmgren, K. Segerberg, & V. Stoltenberg-Hansen, (Eds.), *Logicism, intutionism and formalism: What has become of them?* (pp. 1–25). Springer

Linnebo, Ø., & Pettigrew, R. (2014). Two types of abstraction for structuralism. *Philosophical Quarterly, 64*, 267–283.

Lukasiewicz, J. (1920/1970). O logice trojwartosciowej. *Ruch Filozoficny*, 5: 170–171. English translation in L. Borkowski (Ed.), *Jan Lukasiewicz, Selected Works*. PWN

MacBride, F. (2006). What constitutes the numerical diversity of mathematical objects? *Analysis, 66*(1), 63–69.

Maley, C. (2020). Analog computation and representation. *British Journal for the Philosophy of Science*

Miłkowski, M. (2013). *Explaining the computational mind*. MIT Press.

Miłkowski, M., & Fresco, N. (2019). Mechanistic computational individuation without biting the bullet. *The British Journal for the Philosophy of Science*. https://doi.org/10.1093/bjps/axz005/5305023

Millhouse, T. (2019). A simplicity criterion for physical computation. *The British Journal for the Philosophy of Science, 70*(1), 153–78. https://doi.org/10.1093/bjps/axx046

Papayannopoulos, P., Fresco, N., & Shagrir, O. (forthcoming). On two different kinds of computational indeterminacy. *The Monist*.

Parsons, C. (1990). The structuralist view of mathematical objects. *Synthese, 84*, 303–346.

Parsons, C. (2004). Structuralism and metaphysics. *The Philosophical Quarterly, 54*, 56–77.

Piccinini, G. (2007). Computing mechanisms. *Philosophy of Science, 74*(4), 501–526.

Piccinini, G. (2015). *Physical computation: A mechanistic account*. Oxford University Press.

Pudlák, P. (2013). *Logical foundations of mathematics and computational complexity: A gentle introduction*. Springer.

Rescorla, M. (2013). Against structuralist theories of computational implementation. *British Journal for the Philosophy of Science, 64*, 681–707.

Rescorla, M. (2014). The causal relevance of content to computation. *Philosophy and Phenomenological Research, 88*, 173–208.

Resnik, M. D. (1981). Mathematics as a science of patterns: Ontology and reference. *Noûs, 15*(4), 529–550.

Resnik, M. D. (2018). Non-ontological structuralism. *Philosophia Mathematica, 1093*(10), 1–13.

Schweizer, P. (2019) Computation in physical systems: a normative mapping account. In: D. Berkich, & M. V. d'Alfonso (Eds.), *On the cognitive, ethical, and scientific dimensions of artificial intelligence* (pp. 24–47). Springer.

Shagrir, O. (2001). Content, computation and externalism. *Mind, 110*(438), 369–400.

Shagrir, O. (2018). In defense of the semantic view of computation. *Synthese, 190*, 4083–108. https://doi.org/10.1007/s11229-018-01921-z

Shapiro, S. (1997). *Philosophy of mathematics: Structure and ontology*. Oxford University Press.

Shapiro, S. (2005). Categories, structures, and the Frege-Hilbert controversy: The status of meta-mathematics. *Philosophia Mathematica, 13*(1), 61–77.

Shapiro, S. (2008). Identity, indiscernibility, and ante rem structuralism: The tale of i and $-i$. *Philosophia Mathematica, 16*(3), 285–309.

Sieg, W. (2008). Beyond Hilbert's reach? In S. Lindström, E. Palmgren, K. Segerberg, & V. Stoltenberg-Hansen (Eds.), *Logicism, intutionism and formalism: What has become of them?* Springer

Sieg, W. (2014). The ways of Hilbert's axiomatics: Structural and formal. *Perspectives on Science, 22*(01), 133–157.

Sprevak, M. (2010). Computation, individuation, and the received view on representation. *Studies in History and Philosophy of Science Part A, 41*(3), 260–70.

Wigglesworth, J. (2018). Non-eliminative structuralism, Fregean abstraction, and non-rigid structures. *Erkenntnis, 86*, 113–27.