What is Information Processing? An Analysis Based on Shannon Information and Algorithmic Information[1]

Nir Fresco

Introduction

It is often assumed, particularly in cognitive science discourse, that computation can freely be described as information processing. However, whether processing information is also sufficient for distinguishing computing systems from non-computing systems is another matter. The Encyclopedia of Computer Science (1976, 647) states that 'information processing might, not inaccurately, be defined as "what computers do"'. This definition is put to the test in this paper, dealing with the question whether concrete digital computation (i.e. digital computation as it is actualised in *physical systems*) can be adequately explained solely in terms of information processing based on two quantitative notions of information.

The roots of the conflation of information and computation are in the attempt to explain the mind in the mid-twentieth century. This venture led to a fusion of information-theoretic language, notions of control and Turing machines (TMs). At that time, the information-theoretic language used was based on the Shannon/Wiener cybernetic conception of information. The original ambition of cybernetics was to explain the behaviour of both living organisms and human-engineered machines using a unified theory utilising concepts of control and information (Cordeschi 2004, 186). Norbert Wiener defined a machine as a 'device for converting incoming messages into outgoing messages [… or] as the engineer would say [… it] is a multiple-input multiple-output transducer' (1966, 32). Still, in general, the problems under consideration were based on single-input single-output transducers without any long-range statistical dependencies of transmitted messages.

However, the modern concept of information is broader and need not be limited only to this cybernetic conception. To some degree, everything can be described in information-theoretic terms: from the movements of atoms and molecules through economics, politics and fine arts to ethics and human nature. The concept of information is so broad that some readings of information unavoidably lead to pan-informationalism.

Importantly, a distinction should be drawn between 'data' and 'information'. In the Encyclopedia of Computer Science, data are defined as 'physical symbols used to represent information for storage, communication or processing' (1976, 641). Information, on the other hand, is defined as 'knowledge, especially as it provides people (or machines) with new facts about the real world' (ibid). Data apparently are the vehicle that conveys meaningful, or semantic, information and could be analysed, for example, by Claude Shannon's information theory (SIT) or by algorithmic information theory (AIT) discussed below.

The resulting Information Processing (IP) account hinges on the particular interpretation of information. It can be interpreted semantically, e.g., as factual or instructional information, or non-semantically, for example, as Shannon information (SI) or algorithmic information (AI). To set the stage, the processing of information is characterised here as the production of new information, its modification and its removal. To a first approximation, this characterisation seems to accord with the operations of a TM reading from and writing to a tape and changing states. Also, intuitively, it seems to describe the practical use of information by natural cognitive agents beyond their mere communication of information.

In the short analysis that follows, I discuss some characteristics of resulting IP accounts that are based on either SI or AI. The next section briefly introduces the notions of SI and AI. Section 3 outlines what the processing of SI and AI respectively amounts to in the context of concrete digital computation. Section 4 then examines some of the problems that such IP accounts of computation face. Finally, I conclude that whilst AI may fare better than SI in some regards as a candidate for a plausible IP account of concrete computation, both resulting accounts are inadequate.

Non-semantic notions of information

First candidate: Shannon information (SI)

Shannon (1948, 379) attempted to solve the 'fundamental problem of communication': finding the optimal manner for reproducing messages from a source of information at their destination. According to Wiener (1948, 61), one of the simplest unitary forms of information is the recording of a choice between two equiprobable basic alternatives. 'The significant aspect is that the actual message is one selected from a set of possible messages' (Shannon 1948, 379).

SI does not entail any semantic content or meaning. SIT has abstracted from the physical media of communication (e.g. the physical composition of the communication channel) so that any relevant physical constraints can be analysed separately. Shannon provided a statistical definition of information as well as general theorems about the theoretical lower bounds of bit rates and the capacity of information flow. SIT approaches information quantitatively: how much information is conveyed. SI is different from the ordinary usage of 'information'; it tells us nothing about the usefulness of or interest in a message. The basic idea is coding messages into a binary, or any other, system at the bare minimum number of signals needed to get a message across in the presence of noise.

Even in this sense, the amount of information conveyed is as much a property of our own knowledge as anything in the message. If we send the same message twice, that is, a message and its copy, the information in the two messages is not the sum of that in each. Rather the information only comes from the first one. Receiving a message may change the recipient's circumstance from not knowing what something was to knowing what it is (Feynman 1996, 118–120). The more possible messages a recipient could have otherwise received, the more "surprised" the recipient is when it gets that particular message. The average amount of uncertainty or surprise of the recipient is also known as informational entropy (Floridi 2009, 34).

Second candidate: Algorithmic information (AI)

AIT, which was developed by Andrei Kolmogorov, Ray Solomonoff and Gregory Chaitin, deals with the informational complexity of data structures (but not with the computational complexity of algorithms). It formally defines the complexity or the informational content of a data structure, say, a string, as the length of the shortest self-delimiting program[2] producing that string as output on a universal TM (UTM) (Chaitin 2004, 107). The AI complexity of any computable string (in any particular symbolic representation) is the length of the shortest program that computes it on a particular UTM.

Chaitin proposes thinking of a computing system as a decoding device at the receiving end of a noiseless binary communications channel (2004, 107). Its programs are thought of as code words and the output of the computation as the decoded message. The programs then form what is called a 'prefix-free' set so that successive messages (e.g., procedures) sent across the channel can be distinguished from one another. Despite the different approach to measuring how much information a message carries, Chaitin acknowledges that AI has precisely the formal properties of SI relative to individual messages (2004, 120).

AI may be deemed a competing notion of SI by allowing us to assign complexity values to individual strings and other data types. Like SIT, AIT deals with data encodings of single data structures (typically, strings). Whilst SIT analyses the amount of information of a message relative to a group of messages, AIT analyses the complexity of a string as a single message (Adriaans 2008: 149). The relative frequency of the message (which is the focus of former theory) has no special import. The length of the shortest program producing this message is minimal within an additive constant that encapsulates the size of the particular UTM[3] representing the amount of information in that message (Calude et al. 2011).

Characteristics of the two resulting IP accounts based on SI and AI

An SI-based IP account is, at best, limited in its ability to explain discrete *deterministic* computation in physical systems. Such an account clearly has some merit for explaining concrete computation. SIT emphasises the role of symbol structures as designating a particular state of affairs out of some larger set of possible states (i.e. selective information) (Ralston 1976, 647). SI is fundamentally a non-actualist conception of information, for it considers the actual world as simply one of many other possible worlds. Accordingly, the actual message transmitted is just one of many other possible messages. To a great degree, conventional digital computers are non-actualists, in the sense that they are designed to resist minor perturbations, such as noise, and respond to a broad class of inputs. Being non-actualist suggests some compatibility between SI and digital computing systems.

Indeed, the selective characteristic of SI is compatible with a particular control structure enabling the remarkable flexibility of programmable computing systems. Selective information is closely connected with the way in which digital computers are capable of performing a conditional branch. This operation detects which of several different states obtains (e.g. what input was entered or which symbol was last scanned by the TM) and then sends the computation along different paths accordingly. The use of selective information by conditional branch processes lies at the heart of everything complex that digital computers do (ibid.).

Furthermore, hardware malfunctions in deterministic computing systems could be described as noise in discrete communication channels. SIT deals with those aspects of communication where a transmitted signal is perturbed by noise during transmission. The received signal is then analysed as a function of the transmitted signal and a probabilistic noise variable (Shannon 1948). When considering a miscomputation that is the result of some hardware malfunction, such as a malformed signal, an analysis of it in terms of SI processing can be useful.[4] The computer's memory registers, for instance, are designed to handle such noise by including error correction mechanisms using parity bits, information redundancy, etc.

Yet, SI is a probabilistic concept, whereas digital computation may be either deterministic or not deterministic (e.g. probabilistic computation or pseudo-random computation). The description of a physical system in terms of SI is only adequate if it is memoryless, that is, if the system's transition to state S_{j+1} is unrelated to its last state S_j. But this is typically not the case for most conventional digital computing systems, which are deterministic, for their behaviour is repeatable and systematic.

Whilst the state transitions of Shannon's communication model are probabilistic, the transition probabilities of a *deterministic* TM are all set to 1. For every possible valid input (and a given initial state), there is only one possible state into which the TM transitions. Every future state transition can be accurately predicted by simulating the program being executed. So, in the case of idealised TMs, there is no element of uncertainty or surprise, on which SI is based. Nevertheless, in the case of conventional digital computers, which are susceptible to noise (at both the software and the hardware levels), an SI-based IP account may be useful in describing such potential adverse effects on otherwise deterministic computations.

Furthermore, an SI-based IP account cannot tell us what particular operations are required to solve a particular computational problem. Such an account could be instructive in describing some basic primitive set of operations to solve this problem. SIT provides mathematical measures to calculate the *lower* bounds of information flow along a channel. It can tell us that a solution to a certain problem cannot be computed in less than n bits. But an SI-based analysis cannot distinguish between two equally small circuits or different optimal programs that solve the same problem (and there are infinitely many such programs).

AIT, on the other hand, can distinguish between different optimal programs that solve a specific problem. In principle, the set of all such possible optimal programs is enumerable (Calude 2009, 82). And the full description of each one of these enumerated programs could be provided. However, traditional AIT can only *approximate* the complexity of a string relative to a particular UTM and this prevents us from actually having a full description of all the optimal programs producing that string (Calude et al. 2011). Still, a variation on traditional AIT, which is not based on UTMs, allows us to

compute the complexity of strings (or the exact length of these optimal programs), but this comes at a cost.

Finite State Transducer AIT (hereafter, FSTAIT) relies on finite transducers for which the universality theorem[5] is false. A transducer, in this context, is a finite state automaton with outputs. It can be described as the 6-tuple $(Q, \sum, \prod, q_0, Q_F, E)$, where Q is the finite set of states, $\sum$ and $\prod$ are the finite sets of input and output symbols respectively, $q_0 \in Q$ is the initial state, $Q_F \subseteq Q$ is the set of accepting states and E is the finite set of transitions (ibid.). Since there is no universal transducer, FSTAIT cannot achieve universality.[6] At the same time, the finite state complexity of a string explicitly includes the size of the particular transducer running the program and this complexity becomes computable (whereas traditional AI complexity can only be approximated). For our purposes, this means not only that different optimal programs are enumerable, but also that they are *distinguishable* from one another.

An AI-based analysis is closely coupled to the implementing machine used to run the program in question. The machine's size is included as part of the encoded length of the computed string (Calude et al. 2011). When AIT examines which problems can be solved, either a particular self-delimiting UTM or a specific combination of finite state transducers (in the case of FSTAIT) is considered.

Moreover, in a manner similar to SI, AI complexity is a non-actualist conception of information. In accordance with the universality theorem, any program in some computer language can be converted into a program running on a UTM. There is some algorithmic procedure for passing among the possible enumerated TMs that compute a function *f*. To that end, we can pretend that we have all these enumerated TMs in front of us. If, for instance, we need to compute 20 steps in the computation of the 5th machine on input string 'slam dunk', then we simply pick the 5th machine, put 'slam dunk' on its tape and run it for 20 steps (Downey & Hirschfeldt 2010, 9).

In order to assign AI complexity to the configuration of some computing system (both the machine and the self-delimiting program it simulates), the system is 'frozen' at some point in time. That snapshot of the *actual* computation taking place (rather than all possible counterfactual

computations) is assigned an AI complexity value. Still, since all the enumerated TMs for computing *f* are available in principle, the AI complexity analysis is not strictly limited just to the actual computation.

Problems for the resulting IP accounts

To start with, an IP account of concrete computation that is underpinned by SI has a limited explanatory power concerning effective computability (say, in terms of TM operations). SI only makes sense in the context of a set of potential messages that are communicated between a sender and a receiver and a probability distribution over this set (Adriaans 2008, 146–147). There is no room for a probabilistic selection of messages in describing deterministic processes, for the probability is 1, barring potential adverse effects of noise as discussed above. Deterministic computation is describable as a *specific set* of messages that are selected, encoded and transmitted in a certain ordered sequence of steps, regardless of the probabilities associated with each message. The processing of SI does not entail deterministic digital computation and an SI-based IP account of computation is, therefore, inadequate.

AI is an improvement on SI in some regards as a basis for an IP account of computation. AIT analyses the complexity of a string relative to a particular UTM as a single message and hides the probabilistic message selection process of SIT. AIT (and more specifically FSTAIT) can describe the behaviour of *optimal* programs. FSTAIT can describe the behaviour of non-optimal programs too, if the size of the program in bits is specified (Calude, personal communication). The resulting non-optimal programs become enumerable and distinguishable from one another as in the representative case of optimal programs.

As suggested before, characterising computation as the processing of either SI or AI is problematic. The focus of SI is not on the content of individual messages, but that content is precisely what gets *manipulated*. Processing SI can be the modification of the states of signals encoding the possible messages. It can also be the elimination of possibilities, that is, the reduction of uncertainty, represented by a signal or the introduction of

redundancy to offset the impact of noise and equivocation. But sending the same message twice, as a means of introducing redundancy, does not yield more information than that in each. Similarly, the elimination of redundancy does not reduce the underlying informational content.

Noise on the channel is the source of modification and removal of information and uncertainty is the source of new information. The modification of SI by using error correction methods is a means of offsetting noise. But even then the informational content of the messages remains unmodified. And noise that causes the removal of information is typically physical and rarely ever deliberate. We constantly try to find new ways to minimise the adverse impact of noise on communication (e.g. by using parity check bits or Hamming codes). By contrast, the deletion of information in digital computing systems can be completely deliberate, say, to free up memory resources. As well, new SI is produced only relative to the uncertainty associated with that information. If the entropy of a message in a particular context is 0, then sending this message will not amount to producing new information. Since deterministic computation does not increase uncertainty, it cannot be said to produce new SI.

What does the processing of AI amount to? Producing new information amounts to the system producing an output string SOUTPUT that *encodes* more information than the input string SINPUT. The production of new information amounts to the AI complexity of SOUTPUT being greater than that of SINPUT. Conversely, the deletion of information amounts to the system starting with SINPUT and producing SOUTPUT with *less* information than SINPUT. If data are construed as lack of uniformity, then a complete deletion of *all* data can only be achieved by the elimination of *all* differences, thereby restoring uniformity. A system that produces SOUTPUT with less information than SINPUT means that only some unwanted information is discarded.

Consider for example a selective deletion of certain entries in a database. Once the *place in the memory* holding some data is overwritten, the original data are deleted. For example, the string 'birtdayh happy' may be deleted from the database and be overwritten by 'happy birthday'. This is a typical scenario in classical computing systems. But it differs from the case of information dissemination within the computing system,

say, when parts of the computer's memory are compressed and copied from one register to another, decreasing the system's AI complexity over time by means of self-organisation through data compression and the structuring of unstructured data.

However, strictly speaking, an IP account based on AI will have a limited capacity to explain cases in which information is removed and/or modified whilst the overall informational complexity does not decrease (i.e. when $INF(\text{S}_{\text{INPUT}}) \leq INF(\text{S}_{\text{OUTPUT}})$). Unless the particular UTM (or combination of finite state transducers) running the program is *changed* as a result of removal of information, AIT cannot account for this removal operation.

It seems then that, as expected, the processing of AI cannot be easily decoupled from the underlying computing system running the algorithm or program. The underlying architecture of the computing system and the supported instructions are implicitly included in the calculation of AI complexity. Both conventional AIT and FSTAIT deal with idealised computation. They deal with what happens between input and output whilst *assuming* faultless computation (Calude, personal communication). Any possible errors during the actual computation are ignored. AIT is based on idealised UTMs and FSTAIT is based on faultless transducers. The processing of AI certainly entails deterministic computation, simply because AI is *defined* in terms of some computing system. Nevertheless, an AI-based IP account of computation inevitably yields a circular definition of computation. For AI is defined in terms of computational programs and the very computing systems, which we seek to explain. An AI-based IP account of computation is inadequate, too.

Conclusion

An IP account, on the face of it, seems like a natural and promising candidate for explaining concrete digital computation and for individuating physical computing systems. But it is not as obvious as it first seems. Its explanatory power depends on what we take 'information' to be. Whilst AI may fare better than SI in some regards as a candidate for a plausible

IP account of concrete computation, both resulting accounts are inadequate. Of course, it does not follow that other accounts of computation in terms of information processing are inadequate. In fact, I have argued elsewhere that an account of computation in terms of instructional information processing is adequate for the job.

Notes

1 This paper was presented at the Twelfth Asian Logic conference in Wellington, New Zealand. It is a partial preprint of an article that appears in *Philosophy & Technology* ('Information Processing as an Account of Concrete Digital Computation'; DOI: 10.1007/s13347-011-0061-4). It is reproduced for the 2012 yearbook of the IAS-STS, Graz with the permission of Springer-Verlag. The final publication is available at www.springerlink.com.

2 The domain of each UTM is self-delimited much like programming languages. For they provide constructs that delimit the start and end of programs. A self-delimiting TM does not 'know' in advance how many input symbols suffice to execute the computation (Calude, 2002: 34–35).

3 UTMs differ in implementation resulting in the informative content of a string being relative to the particular UTM used to calculate its AI complexity, K. Cristian Calude shows that for every two UTMs u_1 and u_2 $\forall x \exists c$: $(x \in S, c \in N)$ $|Ku_1(x)–Ku_2(x)| \leq c$ where x is the input to the UTM (and S is the set of all strings) (2002, 8).

4 To a first approximation, a miscomputation is an error in the computation process due to a hardware malfunction or a runtime error of the executed program. Other errors are the result of a poor design introduced by the system's designer or programmer, but, strictly, the computing system should not be "blamed" for such errors that manifest at runtime. Miscomputation is analysed at length in Fresco & Primiero (2013).

5 This theorem states that there exists a self-delimiting UTM *U*, such that for every self-delimiting TM *T*, a constant *c* can be computed (depending only on U and T), satisfying the following property. If T(x) halts, then U(x') = T(x), for some string x' whose length is no longer than the length of the string x plus c (Calude 2009, 81).

6 There is no finite generalised transducer that can simulate a transducer running some program. Yet, Calude et al. (2011) prove that the invariance theorem (informally saying that a UTM provides an optimal means of description up to an

additive constant) also holds true for finite state complexity. Finite state complexity of a finite string x is defined in terms of a finite transducer T and a finite string s such that T on input s outputs x. It is defined relative to the number of states of transducers used for minimal encodings of arbitrary strings.

References

Adriaans, P. (2008), 'Learning and the cooperative computational universe', in P. Adriaans and J. van Benthem (Eds.), *Handbook of the Philosophy of Science, Volume 8: philosophy of information*, Elsevier, 133–167.

Calude, C. S. (2009), 'Information: The algorithmic paradigm', in G. Sommaruga (Ed.), *Formal theories of Information*, Springer.

Calude, C. S., Salomaa K. and Roblot, T. K. (2011), 'Finite state complexity', *Theoretical Computer Science* 412: 5668–5677.

Chaitin, G. J. (2004). *Algorithmic information theory*. Cambridge, UK: Cambridge University Press.

Chaitin, G. J. (2007), *Thinking about Gödel & Turing: Essays on complexity, 1970–2007*, Singapore: World Scientific.

Cordeschi, R. (2004), 'Cybernetics', in L. Floridi (Ed.), *The Blackwell Guide to Philosophy of Computing and Information*, Oxford: Blackwell, 186–196.

Downey, R. G., Hirschfeldt, D. R. (2010), *Algorithmic Randomness and Complexity*, New York: Springer.

Feynman, R. P. (1996), *The Feynman Lectures on Computation*, Reading, MA: Addison-Wesley.

Floridi, L. (2009), 'Philosophical conceptions of information', in G. Sommaruga (Ed.), *Formal Theories of Information*, Berlin: Springer, 13–53.

Floridi, L. (2011), *The Philosophy of Information*, Oxford, UK: Oxford University Press.

Fresco, N., & Primiero, G. (2013). 'Miscomputation'. *Philosophy & Technology*. doi:10.1007/s13347-013-0112-0

Ralston, A. (1976), *Encyclopedia of Computer Science*, New York: Van Nostrand Reinhold Company.

Shannon, C. E. (1948). 'A mathematical theory of communication'. *Bell System Technical Journal*, 27, 379–423, 623–656. doi:10.1145/584091.584093

Turing, A. M. (1950), 'Computing machinery and intelligence', *Mind* LIX: 433–460.

Wiener, N. (1966), *God and Golem, Inc.: A comment on certain points where cybernetics impinges on religion*, Cambridge, MA: The MIT Press.