Science & Philosophy Volume 8(2), 2020 pp. 17-46

Insights in how computer science can be a science

Robert W.P. Luk[♦]

Abstract

Recently, information retrieval is shown to be a science by mapping information retrieval scientific study to scientific study abstracted from physics. The exercise was rather tedious and lengthy. Instead of dealing with the nitty gritty, this paper looks at the insights into how computer science can be made into a science by using that methodology. That is by mapping computer science scientific study to the scientific study abstracted from physics. To show the mapping between computer science and physics, we need to define what is engineering science which computer science belongs to. Some principles and assumptions of engineering science theory are presented. To show computer science is a science, we presented two approaches. Approach 1 considers computer science as simulation of human behaviour similar to the goal of artificial intelligence. Approach 2 is closely related to the actual (scientific) activities in computer science, and this approach considers computer science based on the theory of computation. Finally, we answer some of the common outstanding issues about computer science to convince our reader that computer science is a science.

Keywords: Computer Science, Artificial Intelligence, Theory of Computation, Engineering Science, Science.[1]

♦ *Department of Computing, The Hong Kong Polytechnic University*, Kowloon (Hong Kong). E-mail address: csrluk@comp.polyu.edu.hk

[1] Received on October 18th, 2020. Accepted on December 18th, 2020. Published on December 31st, 2020. doi: 10.23756/sp.v8i2.531. ISSN 2282-7757; eISSN 2282-7765.

1. Introduction

There are many famous scholars who have directly or indirectly thought that computer science is not a science. For example, Frank Harary (Weinburg, 2001) indicated that any subject that has the name, science, in it is guaranteed to be not a science including the subject, computer science. The Nobel Laurate, Richard Feynman (Dlouhy, 2011), indicated that computer science is engineering. He thought that he slipped into engineering from science (as he was a physicist) because he started to work on quantum computing. Abelson (MIT OpenCourseWare, 2009) of MIT gave a lecture indicating that computer science is not about computer and is not a science. He concluded that computer science is a terrible name. More recently, Krebsbach (2015) wrote a paper specifically to say that computer science is not about computers and is not science (in the ordinary sense of the word).

Another camp of this issue considers that computer science is a science, and members of this camp are just as illustrious as the other camp. Herbert Simon who is a Nobel Laureate in economics and an ACM Turing award recipient advances the notion of the sciences of the artificial (Simon, 1969) in which computer science is one such science. Later, Newell (another ACM Turing award recipient) and Simon (1976) consider that computer science as an empirical science (Polak, 2016) based on empirical enquiry like many natural sciences. Their basic logic is that computer scientists are engaged in the study of phenomena about computers and scientific study is about studying phenomena, so computer science is a kind of (empirical) science. Peter Denning, who was the ACM president, does not just advance computer science as science (Denning, 2005) based on general principles (Denning, 2003) but as natural science (Denning, 2007). According to Denning's view, he considers that in natural sciences (like biology), information processing is abundant and that the study of natural information processing in biological systems for example qualifies computer science to be called a natural science.

It seems that most of the proponents that computer science is science are prominent computer scientists while those that deny computer science is a science include computer scientists as well as scholars of other fields like physics and mathematics. To convince all that computer science is a science, it needs to explain why, and this has been done for information retrieval (Luk, 2020). The underlying explanation that information retrieval is a science is because it is like physics which is a well-known science subject. Similarly, to explain why computer science is a science is to show how computer science is like one science subject, say physics, and then claim computer science is science. Since this has been done for information retrieval, our focus on showing computer science is science will focus on how to map computer science to science instead of answering "why" which has been explained before. Luk

(2020) tried to muster as many pieces of evidence as possible to support the notion that information retrieval is a science and this turned out to be a tedious task as well as sometimes obscuring the objective to show that information retrieval is science. Therefore, we take a slightly different methodology by assuming that the reader is familiar with the paper by Luk (2020) and proceeding to highlight how computer science is a science by discussing some of the controversial issues when mapping some of the computer science aspects to science properties in scientific study (Luk, 2017), which are abstracted from physics. In this way, we can keep a clear track of our objective to show computer science is science, and if the reader is in doubt, (s)he can consult the paper by Luk (2020) to investigate in order to further establish that computer science is science.

The rest of this paper is organized as follows. Sect. 2 presents a short literature review about what is computer science. Sect. 3 delineates how we can show a discipline is science in the general case. This section indicates clearly which principles and assumptions are discussed later and which ones are skipped because they may hold in a self-evident way. Sect. 4 presents what is engineering science because computer science is thought to belong to engineering science. Apart from defining what is engineering science, some principles and assumptions are also provided. Sect. 5 presents the two approaches to show computer science is a(n) (engineering) science. The first approach shows computer science is a science by considering computer science as simulation of human behaviour. Next, we present the second approach which considers how the theory of computation can be used to help in showing that computer science is science. This approach reflects better the (scientific) activities in computer science. Sect. 6 discusses some outstanding issues for example why we prefer the name computer science over computing science. Finally, Sect. 7 provides the concluding remarks.

2. Related work

Wegner (1976) cites four influential definitions of computer science at the time. They are "(1) computer science is the study of phenomena related to computers (Newell, Perlis and Simon, 1967), (2) computer science is the study of algorithms (Knuth, 1968), (3) computer science is the study of information structures (Wegner, 1968) and (4) computer science is the study and management of complexity (attributed to Dijsktra by Wegner, 1976)". Wegner (1976) boils these four definitions down to three traditions of computer science, corresponding to three different periods in computer science. The three traditions are empirical tradition exemplified by definition (1), the mathematical tradition (McCarthy, 1962; Knuth, 1974) exemplified by definitions (2) and (3),

and the engineering tradition exemplified by definition (4). Eden (2007) coins these three traditions as paradigms: the rationalist paradigm (corresponding to the mathematical tradition), the technocratic paradigm (corresponding to the engineering tradition) and the science paradigm (corresponding to the science tradition). It is thought that all three paradigms or traditions exist in computer science so that it is difficult to classify computer science into some existing discipline.

Recently, Rapaport (2017) tries to tackle the question, what is computer science, by surveying various ways computer science can be defined or described, as well as trying to develop his own way of defining computer science discipline. This leads him to consider that computer science may belongs to a new type of engineering or a new type of science or an exclusive-or of these two general disciplines. However, the exact nature of this new type of engineering or science is unknown or not described in full by him, even though we are a kind of concur with him as we develop the (new) discipline of engineering science. He concludes that "our exploration of the various answers suggests that there is no simple, one-sentence answer to our question (i.e., what is computer science). Any attempt at one is no better than the celebrated descriptions of an elephant by the blind men" (Rapaport, 2017).

Perhaps, the definition of computer science in the book about algorithms and data structures by Miller and Ranum (2015) is close to our definition of computer science. Miller and Ranum (2015) consider that "computer science is the study of problems, problem-solving, and the solutions that come out of the problem-solving process". However, they do not mention that the problem-solving process involves a programmable device as our definition of computer science requires, and they do not further develop a framework of understanding computer science based on problem solving as in this paper. Also, they quickly turn their attention to algorithms and they proceeded to consider that "computer science can be thought of as the study of algorithms", which concur with one of the influential definitions of computer science. Therefore, even though Miller and Ranum mention that computer science is related to problem solving, they do not develop this idea more fully as in this paper. This is similar to Margolis and her colleagues in 2008 who quote a "spot-on" definition of computer science from a users' guide for Stanford University computer science majors, i.e., computer science was "the science of solving problems with the aid of a computer" (Margolis et al., 2008). Again, there is no further elaboration by Margolis et al. about this definition, so there is no framework for this understanding of computer science as a scientific discipline.

3. Methodology to show a discipline is science

Rapaport (2020) spends about several hundred pages to discuss philosophy of computer science. In here, we cannot spend a similar number of pages to convince people that computer science is science. Instead, we only highlight the important aspects that makes computer science a science and leave out the nitty gritty to the reader to fill in the missing link himself or herself if (s)he is in doubt. Therefore, this paper focuses on the insights that we can gain from making computer science a science rather than showing the evidence to support the claim.

We focus our description of computer science on the computer and programming/computation as these are thought to be the shared aspects in computer science. For example, software engineering is about how programs are written so this is related to our notion of computer science. Another example is in the applications of computer science in which computers are used to solve problems for the user, so this relates to our notion of computer science. Other examples include interests in computational complexity where the efficiency of the programs or algorithms is analysed, which is related to our notion of computer science. However, we do not divert our attention to human-computer interface even though it is related to our notion of computer science as how computers present information to users for effective and efficient communication is important in computer science. This is because interface is not thought to be the core shared part of computer science, which does not affect our claim that computer science is science, so we will not discuss it in here.

Our methodology to show that a discipline is science is based on the work on showing information retrieval is science by Luk (2020). First, we try to define what is the aim of the concerned area of study based on instantiating the aim of scientific study in the context of that study. In this way, we establish that the aims of the various science disciplines are similar to each other and the difference is only that the aim is applied in the context of the particular science discipline. This helps to establish the unity of the different scientific disciplines.

Second, our methodology tries to show that computer science is a mature science (Luk, 2010). Therefore, we need to show that computer science has a framework of theory, model, experiment and physical situation arranged in some kind of hierarchy with inter-connections. This framework is important for mapping the computer science knowledge to knowledge in other scientific disciplines. It is important as these are shared commonalities between different scientific subjects that enable the different subjects to claim that they belong to science.

Third, the principles (Luk, 2017) of scientific study (Table 1) are applied to engineering science and/or computer science. Since some of these principles are obviously applicable, we will not discuss them further in this paper. For

example, the principle of immutable laws and principles is applicable to engineering science as the laws and principles are supposed to be unchanged once formulated in engineering science and computer science.

No.	Principle Name	Discussed Here	No.	Assumption Name	Discussed Here
1	Generalization	Sect. 5.1.2 & Sect. 5.2	1	Sufficiently trained	No
2	Modelling accuracy	Sect. 5.1.2 & Sect. 5.2	2	Accurate communication	No
3	Empiricism	Sect. 5.1.2 & Sect. 5.2	3	Unbiased, accurate observation	No
4	Theoretical objectivity	No	4	Adoption of the aim of scientific study	No
5	Theoretical consistency	No	5	Causality of phenomenon	Sect. 5.1.2 & Sect. 5.2
6	Immutable laws and principles	No	6	Explanatory power	No
7	Objective experiment	No	7	No magic	Sect. 5.1.2 & Sect. 5.2
8	Reliability	No			
9	Investigation objectivity	Sect. 5			

Table 1: The principles and assumptions of scientific study (as detailed in [Luk, 2017] and as mentioned in [Luk, 2020]) with the indication on whether they are discussed in this paper about their applicability to computer science.

Fourth, the assumptions (Luk, 2017) of scientific study (Table 1) are assumed to hold for engineering science and computer science. Again, we have indicated in Table 1 which assumptions are obviously applicable and so we will not discuss their applicability in engineering science and computer science. For example, the engineering scientists and computer scientists are obviously assumed to be sufficiently trained to carry out the scientific investigations, so it is not necessary for us to discuss whether assumption 1 (in Table 1) is applicable to engineering science or computer science.

Fifth, when Luk (2020) shows that information retrieval is science, he cited papers relating to the different activities in the interaction model of scientific study (Figure 1 of [Luk, 2020]). In here, we do not make this kind of citations as we feel that it is fairly self-evident that engineering science and computer

science follows the interaction model of scientific study. If in doubt, one can consider information retrieval as a sub-discipline of computer science, and the citations made in Luk (2020) can be considered as supporting evidence that computer science and therefore engineering science (because information retrieval is a branch of computer science which in turn is a branch of engineering science) follows the interaction model of scientific study.

4. Engineering science

To answer the question what is engineering science (see [Boon, 2008] for some background), we need to know what engineering is. Engineering can be considered as a problem-solving activity. However, it is not any type of problem-solving activity but that it involves (a) a technical problem and (b) using a device to solve a problem. Therefore, we can describe engineering as using a man-made device to solve a technical problem. So, how can such a discipline or its sub-discipline be considered as a science?

4.1 Engineering science as applied science

One way that engineering can be considered as a science is that in the technical-problem solving activity, it made use of scientific knowledge to solve the problem. Effectively, some aspect of engineering science is considered as an applied science where scientific knowledge is applied to solve technical problems. For example, in mechanical engineering, when Newton's laws of motion are used to solve some technical mechanical problems, then we would consider that this type of problem-solving activity as mechanical engineering science. Another example is in electrical engineering where ohm's law is used to calculate the voltage or current of an electrical device like the light bulb. This kind of problem-solving activity can be considered as electrical engineering science.

4.2 Engineering science as pure science

Another type of engineering science is that it is similar to pure science. What is pure science? Here, we follow Luk's idea (Luk, 2010; 2017) that pure science has a knowledge structure (called a framework) similar to physics with theory, model, experiment and physical situation (in a hierarchy). Therefore, for some engineering science to be a pure science, the knowledge of the engineering science needs to be arranged into a hierarchy of theory, model, experiment and physical situation. The theory would contain the principles which are applied to build models, the predictions of which are measured in experiments. For

example, the probability theory of information retrieval (Luk, 2020) has the probability ranking principle (PRP), which is applied to build retrieval models based on TF-IDF term weights, the prediction of which is specified by the PRP (Luk, 2020). The prediction error is the optimal accuracy specified by PRP minus the actual accuracy of the retrieval model. The actual accuracy is measured in terms of the recall and precision of the ranked list produced by the retrieval model. Therefore, we can think of information retrieval as a (pure) engineering science.

In engineering science in general, the prediction is about whether the device can solve the problem in the problem-solving activity. Therefore, the prediction accuracy is about the prediction of solving the problem. Typically, we assume that the device can solve the problem. So, this is the assumption in engineering science which we call the *universal solvability assumption* in engineering science theory. If we make such an assumption, then we predict that the problem is solved (i.e., 100% solvability) by our device. If we can only solve it partially say 25%, then the prediction error is what we predicted minus 25% (i.e., 100% - 25% = 75%). Furthermore, better devices are those that have better prediction errors or those that are closer to the universal solvability assumption. By formulating the prediction in this way, better devices are similar to better scientific models that have better prediction accuracy (or less prediction errors) so that making better devices is similar to building better scientific models, conforming to the aim of scientific study that tries to obtain good quality scientific knowledge (e.g., highly accurate scientific model). Therefore, in this sense, engineering science is like a science.

Engineering science theory contains several general principles. Some of them are related to problem solving activity since engineering science is about problem solving. Specifically, in order to know what the problem is, information needs to be gathered. Gathering information is equivalent to reducing the uncertainty of constructing the device (according to information theory [Shannon, 1948]). So, our most general principle in engineering science is the *minimum uncertainty principle* which states that a problem should be solved with as minimum uncertainty as possible. This principle was identified by Klir (Balsamo et al., 2000) and his co-workers (Klir and Wierman, 1999), but it has not been regarded as the most general principle before. It is rephrased as the above to adapt to engineering science.

While the minimum uncertainty principle guides the gathering of information, it is necessary to know what kind of information to collect. In general, the user has encountered some problem so that the user wants to make use of the device to solve his/her problem. Therefore, the first step is to gather information about the problem. Gathering information about the problem also helps us to gather information about solving the problem. So, the purpose of gathering information is to help us to solve problems, in order to lead to success.

Therefore, our next principle is about the identification of success, as the information gathered tells us how to become successful. The *success identification principle* states that formulating the right problem to be solved by a cost-effective solution is a step towards success. Here, the success of the application is based on solving the problem of the user. Notice that it is not any problem of the user but the "right" problem of the user. The word "right" means that the problem is not related to some superficial problem or epiphenomenon. Instead, this should be a real problem experienced by the user. Because the real problem is not just some superficial problem that can be easily identified, the information gathering process for the problem may need to take some time to find out the real problem of the user, who may not be able to articulate the problem to the engineering scientists. The "real" problem may be some sub-problem of the original problem rather than all the sub-problems, as some of the sub-problems may be insignificant. In this case, identifying the right sub-problem to solve is critical in the problem-solving process. The success identification principle would lead us to the universal solvability assumption since we have a cost-effective solution that can solve the problem, so that the predicted cost-effectiveness is 100%. Given the universal solvability assumption, why do we still want a cost-effective solution? As allured earlier, we are solving the right problem of the user and typically the user is concerned about the cost-effectiveness of the solution as a solved problem that takes a million years to solve is not useful to many users. Therefore, the success identification principle asserts that the predicted cost-effectiveness of the solution should be 100% (if we have the ideal solution for the user).

There is a concern whether the engineering science knowledge is testable since scientific knowledge is testable. For example, it is assumed in the universal solvability assumption that a problem can be solved. For some problems, one can develop an algorithm or solution that guarantees the problem is solved so that the assumption is guaranteed to be fulfilled. Therefore, for some problems, the engineering science theory may not be testable. Note that we have a success identification principle in the engineering science theory. This principle predicts that we can obtain a solution that has 100% cost-effectiveness. This prediction is required because we want to change the prediction accuracy into a predictor of problem-solving ability so that the higher the prediction accuracy the higher the problem-solving ability. Therefore, if we can design our cost-effectiveness measure to be normalized between zero and one, then we can make a man-made device to solve the problem with X% cost-effectiveness. Since the prediction of the cost-effectiveness by the success identification principle is 100%, the prediction error of the man-made device is 100%-X%. Therefore, the higher the cost-effectiveness of the man-made device is, the lower the prediction error and the higher the prediction accuracy of the cost-effectiveness. Why do we focus on predicting the cost-effectiveness instead of the solvability of the device, this

is because some device can guarantee to solve a problem but it may take a long time or use an unimaginable amount of resources (e.g., storage). By requiring the solution to be cost-effective, those solutions that guarantee to solve the problems may not have high cost-effectiveness even though their solvability is 100%. In this way, the success identification principle is testable since we need to build the device, measure its cost-effectiveness before we can say that it has 100% cost-effectiveness as predicted by the success identification principle. As a result, the principle of empiricism is upheld (Table 1).

In engineering science theory, we also have the *no-garbage-in principle*. This principle is originated from garbage-in-garbage-out. The idea is that we should not input garbage into our device because it would produce garbage output that would not solve our problem which would contradict the universal solvability assumption. Therefore, we apply the no-garbage-in principle so that we do not feed garbage into our device when solving our technical problem. This principle is universal meaning that it is applied to any device for any problem.

An example of pure engineering science subject is information retrieval (Luk, 2020). Basically, information retrieval is an engineering science discipline (Fuhr, 2012) which makes use of a device to find documents from a collection. It is like a pure engineering science discipline as discussed in the paper by Luk (2020) where the universal solvability assumption predicts the retrieval accuracy is 100% instead of the probability ranking principle. This can ensure even non probabilistic retrieval models can be included in the prediction of retrieval accuracy so that more models can relate retrieval accuracy with prediction accuracy. Consequently, the retrieval performance is related to one aspect (i.e., accuracy) of the scientific knowledge (i.e., the retrieval model). Another example of pure engineering science is computer science which we are going to discuss in Sect. 5.

4.3 The aim of engineering science

What is the aim of engineering science? It should be similar to the aim of science or the aim of scientific study. Therefore, borrowing from the aim of scientific study (Luk, 2017), we state:

> **Definition**: the aim of engineering science is (i) to produce good quality, general, objective, testable, complete scientific knowledge (as defined in [Luk, 2010]) of technical problem-solving using a man-made device (model) to solve the problem, and to (ii) monitor/apply such knowledge.

Note that the man-made device usually has a model. In fact, most engineering science starts with designing the model of the man-made device first before the physical device is built. So, the conceptual problem solving is done using the device model whereas the physical problem solving is done by the physical device. The aim is about the scientific knowledge of technical-problem solving.

Since the device is used to solve the problem, knowledge of problem solving involves in understanding technically the device on how the problem is being solved with it. This scientific knowledge needs to be organized into theory, model, experiment and physical situation similar to a scientific discipline like physics.

The quality of scientific knowledge needs to be measured in terms of accuracy, reliability, consistency, etc. While reliability and consistency are relatively easy to follow for engineering science, getting more accurate results do not directly imply getting better solutions. Therefore, in the previous section, we formulated the universal solvability assumption so that more accurate solution implies more effective solution, thus establishing a direct link between building better models to constructing better devices in solving problems. An engineering science should also look for general, objective scientific knowledge that is testable. Therefore, engineering science should look for general scientific knowledge (like principles) rather than a set of facts. Engineering science should also disseminate its findings so that the scientific knowledge is shared for objectivity. Engineering science should make devices that are testable so that it has some relations to physical reality. Therefore, engineering science has an aim that is similar to the aim of scientific study. Note that this aim belongs to the engineering science theory according to Luk (see Figure 3 in [Luk, 2017]).

Some scientists (e.g., Feynman in [Dlouhy, 2011]) may object that the engineering science is about using a man-made device to solve problems because the study is about the man-made device which is a human artifact instead of natural phenomena. However, if we use such a criterion to demarcate science and non-science, then what usefulness does it serve to demarcate science in this way. Demarcating science in this way does not make the science subjects to be exact science (Luk, 2018), so such demarcation criterion does not have power over the capability of science. Put it in another way, why cannot the study methods of science be applied to study man-made devices? What important reasons that have repercussion on the power of the study are there to prevent the application of scientific study to man-made devices? We feel that there are no strong reasons to stop applying scientific study methods to other seemingly non-science subjects.

5. Computer science as engineering science

In general, we consider computer science to belongs to engineering science. So, computer science is about solving (technical) problems using programmable devices. However, the device for computer science is programmable. Here, programmable means that the device follows a sequence of instructions to execute its actions and this sequence is stored in some (memory or configured)

device so that if another sequence of instruction is loaded into the (memory or configured) device, then the overall device will execute a different sequence of instructions. Now, the device can only follow a finite variety of instructions, so that the sequence of instructions must be specified to the instructions that the device can understand. Otherwise, the device cannot turn the instructions into meaningful actions for solving the problem.

Since computer science belongs to engineering science, computer science inherits the principles and assumptions of engineering science. For example, the universal solvability assumption in engineering science is inherited by computer science. Therefore, we expect that the computer [Rapaport, 2018] (a programmable device) can solve the problem either completely or partially. Computer science also inherits the no-garbage-in principle in engineering science so that we expect valid, useful inputs are entered into the programmable device. The aim of computer science can be considered as a specialization of the aim of engineering science as follows:

> **Definition**: The aim of computer science is (i) to produce good quality (measured for example by accuracy, reliability and consistency), objective, general, testable, complete scientific knowledge (as defined in [Luk 2010]) of technical-problem solving using a programmable device (model) to solve the problem, and (ii) to apply/monitor such knowledge. (Adapted from [Luk 2017])

The above definition of the aim of computer science is almost the same as engineering science apart from the fact that man-made device (model) is replaced with a programmable device (model). It is implicitly assumed that the programmable device is man-made, so that the aim of computer science is a specialization of an engineering science. As indicated earlier in a similar way, this definition belongs (Luk, 2017) to the computer science theory. Also, the scientific knowledge is about technical-problem solving that involves the technical understanding of the programmable device on how to solve the problem with it.

Investigations in computer science can be objectively done as computer science papers are published in journals, books and conference proceedings. However, for military applications and for commercial applications, the investigation may be held as a secret so that this hinders the objectivity of the investigation. For commercial applications, the results and knowledge of the investigation may be published in patents so that they are disseminated to the public but protected as intellectual properties. There is also a growing trend to have open data sets like UCI machine learning data sets as well as sharing open-source research software like GitHub to ensure investigation objectivity is being upheld. Overall, we believe that the investigation objectivity principle (Table 1)

is applicable to computer science (for approach 1 in Sect 5.1 and for approach 2 in Sect. 5.2).

5.1 Approach 1: Computer science as simulation of human behaviour

While computer science is a way to solve a problem, it is also a model of how a problem is being solved. When we talk about computer science modelling something, that something must correspond to some physical situation. So, the physical situation involves a human agent taking information from a user. The agent tries to solve user's problem by following the instructions given by an instructor. When the problem is solved, the agent gives the solution back to the user. Then, this physical problem-solving activity is complete.

In computer science, we replace the agent by a machine which is a programmable device. The instructor is given a special name called programmer, in computer science. The user remains the same. Effectively, computer science is about using a programmable device to simulate the human agent in following the instructions of the instructor when solving a problem. This perspective of computer science, which is called the agent perspective, is consistent with the view that the Turing machine is simulating a clerk (Anguera et al., 2020) following instructions from a mathematician to solve a mathematical problem like calculating the logarithm of a number before the invention of the calculator. Since simulation is a kind of scientific activity, computing is therefore implicated to relate to science.

5.1.1 Simulation of human behaviour

The science in computer science therefore is in the simulation of human behaviour. If the machine can follow the instructions given by the instructor exactly as the human agent, then we have 100% accuracy of simulation. Therefore, one may be tempted to conclude that computer science is an exact science. However, we have not specified what kind of human instructions we have in mind. For some instructions like writing a symbol on a piece of paper, the machine can simulate the human instructions exactly. For other more high-level instructions like recognizing the human face from a photograph, the machine performing face recognition may not be able to yield 100% simulation accuracy. Therefore, whether the machine can solve a problem with 100% simulation accuracy depends on the kind of instructions that we give to the agent or machine. This perspective ties in with the subject, Artificial Intelligence, in which the goal of Artificial Intelligence can be thought of as the simulation of human behaviour in executing a high-level intelligent instruction. Note that we have to distinguish between simulation accuracy and problem-solving effectiveness as the simulation accuracy can be 100% but the problem solving

effectiveness may be as low as 60% with 100% simulation accuracy because human problem solving effectiveness may be limited to 60%.

In computer science, we usually limit the instructions of the programmable device to simple instructions like moving a symbol to a tape or read a symbol from the input. We rarely specify a high-level instruction. Such a high-level instruction is being broken down into a sequence of finer instructions to solve the problem. Finer instructions are further specified into finer instructions until those instructions are simple enough that they can be understood by the programmable device. Such reduction is at the heart of computer science because computer science assumes that all instructions can be reduced into a sequence of simple instructions. We can formulate this as the *reduction assumption*. This reduction assumption is similar to the universal solvability assumption in engineering science because the reduction assumption like the universal solvability assumption may be solved completely or only partially. Because any human behaviour can be considered as a high-level instruction, computer science can be thought of as the simulation of human behaviour in general.

5.1.2 Is simulation science?

One concern is that while simulation is a scientific activity, it is not clear whether doing simulation implies that the subject is science. One can think of simulation as having a model and based on the model we add details to make the simulation to be part of the experiment for making observations. Those details may not be scientific, for example coding certain aspect of the simulation based on some rule of thumb or heuristic. However, usually the model is scientific based on some theory. For computer science, do we have some theory that constructs the model which is used for developing say a program that performs simulation in our experiments as in mature science like physics (i.e., do we have a framework like physics)?

For computer science, we can rely on the theory of engineering science to form the basis of our model for simulation (Figure 1). First, the theory of engineering science has the aim of engineering science which is specialized to the aim of computer science by replacing the man-made device (model) with the programmable device (model). When solving the problem conceptually, it is usually assumed that we use a programmable device model so solve the problem rather than the actual physical device because it is easier to comprehend and apply the device model than the physical device. A specialization of the programmable device model is the human behaviour model (Figure 1) that models the human behaviour following the instructions of some model of the procedure to solve the problem. The model of the procedure or procedure model gets rid of the minor procedure details so that the human behaviour model can focus on the high-level procedure model for solving the problem. Both human

behaviour model and the procedure model exist in the (scientific) model realm because they do not contain all the details about solving the problem but only sufficient amount of high-level details on how to solve the problem. For example, some details of the programmable device do not belong to the human behaviour model because they may be heuristics for performing the simulation and these heuristics cannot be explained at the model level so that such heuristics do not have problem solving value or scientific value. The human behaviour model is realized by a physical programmable device to perform the simulation of human behaviour in following the instructions in the procedure model. The procedure model is compiled into a detailed procedure for the physical programmable device to follow directly so that the physical programmable device can be thought of as simulating the human following the high-level instructions in solving the problem. The physical programmable device interacts with the physical situation. On the one hand, the physical programmable device extracts the problem information from the physical problem situation or physical state. On the other hand, the physical programmable device executes actions that may change the physical problem situation or state from one to the other until the physical programmable device halts or the problem is solved (i.e., the goal physical state is reached). This mirrors how the human tries to solve a problem by interacting with reality described by different physical states. Therefore, the simulation of human behaviour that follows the instruction to solve problems can be put in a framework of theory, model, experiment and physical situation as in mature science.

As one aspect of the quality of (scientific) knowledge is reliability, we are concerned about the reliability of the simulation of human behaviour since we are claiming that the simulation is part of science. If the instructions are very simple and easy to simulate, it is supposed that the reliability of the simulation is very high. However, when the instructions involve intelligent human behaviour (such as face recognition), the reliability of simulating following these human instructions may not be necessarily high, so measurement of the reliability of the simulation is necessary. For example, machine learning research (e.g., Krueger et al., 2015) and pattern recognition research typically performs cross-validations, reporting the performance with statistical significance as an indication of the reliability of the results.

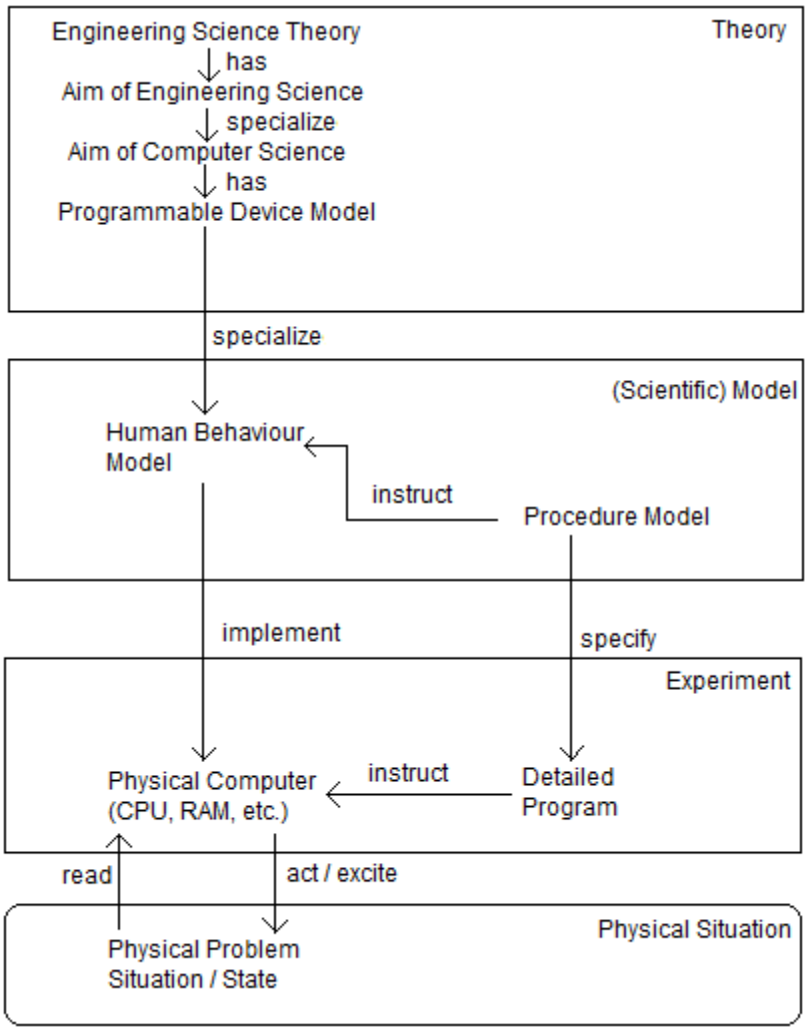

Figure 1: The framework showing theory, model, experiment and physical situation interlinked similar to a mature science like physics (Luk, 2020) for computer science to be simulating human problem solving by following (human) instructions.

Another aspect of the quality of (scientific) knowledge is consistency. According to this approach, it is about the scientific knowledge of the simulation that needs to be consistent. For complex human instructions, their simulations may result in unforeseen inconsistencies in the simulation, which is possible as the simulation is complex. The inconsistencies may not be apparent until the simulation reaches those inconsistencies in the program. As a result, these inconsistencies may be bugs in the program that need to be fixed. Detecting these bugs may not be easy because they may still enable the computer to run,

but they may cause the system to crash later, produce incomprehensible results, or output the wrong results (e.g., calculations).

Note that the framework of theory, model, experiment and physical situation is some kind of hierarchy because theory is more general than model, model is more general than experiment, experiment is more general than physical situations. In computer science based on the notion of simulating human behaviour, the framework is also a hierarchy. While we may have the same aim as in the computer science theory, we may have different procedure models for solving different problem types. The procedure model can be implemented in different programming languages in the experiment level. These different programmes may run on different programmable devices (e.g., different CPUs) so that these correspond to different physical situations. Therefore, we may consider that the generalization principle (see Table 1 and [Luk, 2017]) is upheld in computer science.

One problem with this approach is that the simulation accuracy is not linked with the cost-effectiveness of problem solving so that one cannot claim immediately that the modelling accuracy principle (Table 1) is upheld. The reason is that even if the simulation accuracy is 100%, there is no guarantee that the specified procedure model can solve the problem and therefore the cost-effectiveness is decoupled from the simulation accuracy. Having said that, we expect that most procedure models can solve the problems better than random guess as required by the modelling accuracy principle. The fundamental limit of solvability of this approach is the limit set by human following the instructions of the procedure to model to solve the problem. However, this is not the fundamental limit of the cost-effectiveness of the approach since there may be more cost-effective procedure models when the cost of problem solving is taken into account. Therefore, the cost-effectiveness cannot be guaranteed to be 100% and the success identification principle is testable, so the principle of empiricism (Table 1) is applicable to this approach.

The causality of phenomenon assumption (Table 1) is followed by computer science as human behaviour simulation. This is because the phenomenon that is being studied is the human behaviour. This human behaviour follows a causal chain of actions and reading information based on following the procedure model. Therefore, the simulation of human behaviour following the instructions is also following a causal chain of reading information and executing action like the human. For some simulations, the simulation of the human behaviour may cause some non-determinism. For example, the human instruction may throw a dice and execute according to the number shown on the top face of the dice. If the computer follows this instruction, then there is the uncertainty whether the dice thrown will result in the same face as the dice thrown by human. If they are different, then the simulation may be quite different for the machine compared with the human since the instructions followed by the human and machine may

be different. We assume that the dice thrown by the human and the machine produces the same result so that both follow the same sequence of instructions for the (exact) simulation to take place. Therefore, the assumption about causality of the phenomenon is followed.

The no magic assumption (Table 1) states that if identical or similar situation occurs, then identical or similar distributions of outcome are produced. For this approach, the simulation of human behaviour is the same as the human behaviour if the simulation started with the same initial state and the same input is given to the computer as the human. Therefore, we would expect that the same result state would be arrived. Note that as indicated earlier, there are some complications when the human instructions follow the result of throwing a dice or using a random number in which case the simulation may not result in the same outcome. Therefore, it is assumed in the simulation that the dice outcome or random number outcome of the simulation is the same as that of the human instruction of throwing a dice or using a random number. Hence, we believe that the no magic assumption is followed in computer science as human behaviour simulation.

5.2 Approach 2: Computer science with the theory of computation

In the theory of computation, the instructions are required to be specified in definite ways. Therefore, the concept of "effective procedure" was raised so that we are not talking any kind of instructions that can be performed by human. In this way of thinking, the instructions must be definite and simple enough to be executed by a machine that simulates the human, effectively guaranteeing that the simulation accuracy is 100%. In this light, an algorithm can be thought of as follows:

> **Definition**: An algorithm is a model of a procedure, the high-level instructions (or their equivalent) of which are definite enough to be implementable by a machine (to possibly guarantee 100% accuracy for simulating a human that follows these instructions to solve a problem).

Some prominent computer scientists (e.g., Knuth, 1968) claim that computer science is about algorithms. According to our definition of algorithm, if we concur with this claim about computer science, it implies that we must place some restrictions on the kind of instructions that the problem-solving activity can have. In other words, a procedure or its abstraction is an algorithm as long as we are certain that the high-level instructions (or their equivalent) can be implemented in a machine (which can simulate the human being following such instructions). The advantage of complying with this definition of algorithm is that we can now import the theory of computation as a focused theory of the engineering science theory.

According to the aim of computer science, we are using a programmable device called a computer to solve the problem (see Figure 2). This device has a finite set of instructions that we can specify. These instructions in turn specify the instructions that can appear in the algorithm. Note that an instruction in an algorithm can be a sequence of instructions of the computer so that an algorithm can be any abstraction of an effective procedure that can be implementable on the computer as sequences of computer instructions. Therefore, the engineering science theory specifies a computer, the allowable instructions of which specify the kind of algorithm that we can design. Now, according to the theory of computation, the Turing machine or Turing computer (Rapaport, 2018) is the most powerful among other computing machines (e.g., push down automaton or finite state automaton). In fact, what is conjectured is that what is computable is accomplishable by a Turing computer according to the Church-Turing thesis. Although this thesis was regarded by some as a hypothesis, the Church's thesis is being axiomatized (Gurevich, 2000; Dershowitz and Gurevich, 2008), and the thesis can be derived from four postulates now. If we assume that the postulates for the Church's thesis and the Turing's thesis (Dershowitz and Gurevich, 2008) to hold, then the Church's thesis and the Turing's thesis are logical consequences (i.e., theorems) and the Turing computers can compute computable functions. Turing computers are used to solve problems because they are the most powerful, so our inability to solve a problem is not limited by the capability of the machine or programmable device. Since lambda-calculus is equivalent to Turing computer, one can write algorithms in lambda-calculus to solve problems, and then translate it to run on a Turing machine. However, modern computer scientists do not use lambda-calculus for specifying algorithms. Instead, some high-level specifications are used where the instructions are thought to be implementable in some high-level programming language like Pascal or modulo 2, because these high-level programming languages are like natural languages which are easier for the programmer to write and understand. This can be done because the high-level programming languages are usually Turing complete (i.e., as powerful as the Turing computer) so that programs in these high-level languages can be implemented in some Turing computers for solving problems. These high-level programming languages typically form the basis to specify the equivalent instructions that appear in an algorithm because the programmer or instructor does not want to get into the details of the program that may side-track the problem-solving activity.

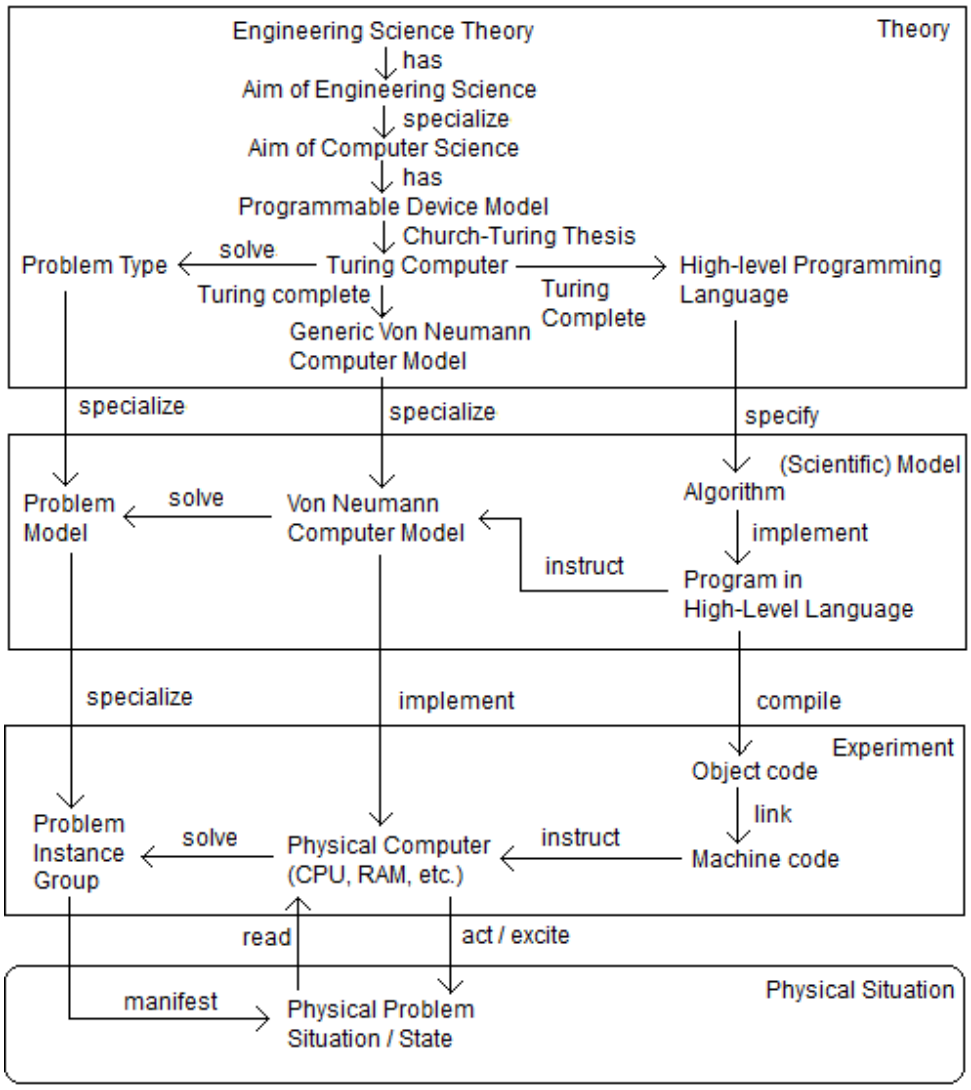

Figure 2: A framework of theory, model, experiment and physical situation applied to computer science with the theory of computation, as problem solving activities. Note that there are other connections between the theory, model and experiment realms. For example, the (engineering science) theory has the success identification principle (not shown) which is applied to the solution (in the model realm) consisting of the von Neumann computer model and the program written in high-level programming language. The solution model then predicts that the physical solution has 100% cost-effectiveness performance according to the success identification principle. The reason why these are not shown in the figure is that it would complicate the figure and make it hard to discern. Instead of showing all the connections, we have selected the ones that show the knowledge can be arranged in a hierarchy framework for illustration purposes.

For those programmers who need to implement the system, they convert the algorithm into programs specified in the high-level programming language. Because the high-level programming language is more or less independent of the programmable device, we regard these high-level programs to be models. When these programs in high-level programming language are compiled into object codes, such object codes are considered to belong to the experiment as these object codes are typically machine dependent. The object codes are still general in the sense for example that the actual addresses of the program need to be recalculated as parameters of the program. When the machine code is generated from the object code, it can be used to execute the physical computer which is usually an implementation of a von Neumann computer model (with some extra functions). Such a model is a derivative of a general von Neumann computer which is found to be Turing complete (Moore, 2014). This is necessary because we need to make sure that the (Turing complete) algorithm that we specify is implementable on a Turing complete machine so that we would not find for some instructions, we cannot specify them for the physical computers.

According to this approach, computer science is scientific in two senses. In one sense, computer science is the simulation of human behaviour similar to Approach 1. However, the instructions in this approach are so simple that the simulation accuracy of the human behaviour following the instructions is 100%. Since we have discussed the simulation of human behaviour in Approach 1, this sense of computer science being a science is not elaborated further in here.

Another sense that computer science is science in this approach (i.e., Approach 2) is that the problem-solving activity is scientific. The problem-solving activity is placed in a framework similar to mature science (Luk, 2010) as in Figure 2. In mature science, we expect that the principles in the theory are applied to build the models which predict the outcome with certain performance in the experiment (Luk, 2010; 2017). Similarly, in computer science (Figure 2), the engineering science theory has the success identification principle which is applied to build the problem model and the solution model consisting of the von Neumann computer model and the program in high level programming language. The solution model predicts the outcome in the experiment and the prediction of the cost-effectiveness performance is 100%. Therefore, the prediction error of the cost-effectiveness is 100% minus the actual cost-effectiveness of the physical solution model. In Figure 2, we have not shown the details of these connections to avoid complicating the figure, which may make it hard to discern the hierarchy framework for illustration purposes.

The problem-solving activity (Figure 2) is being modelled with an algorithm solving a problem model which is a general description of the physical problem. For example, the problem model may specify the problem size as a variable, but the particular problem specifies the problem size with a definite number. The

problem-solving model is implemented in the experiment as a specific instance group of problem solving using a machine running machine codes for a particular problem instance group at hand. The particular problem instance group is manifested in the physical problem situation or state in the physical situation realm. The physical computer (or machine) interacts with the physical problem situation and turns it into a physical goal state that satisfies the user. This corresponds to the physical problem-solving activity that is being modelled by the problem-solving model in the framework. Therefore, the specialization of the aim of computer science produces a problem-solving generic model in the theory realm specifying the problem type that the Turing machine solves. This generic model is then specialized into a problem-solving activity model with specific problem model solved by a von Neuman computer model running some algorithm or high-level language program in the (scientific) model realm. This problem-solving activity model is then specialized to a particular problem instance group solved by a machine running machine codes in the experiment realm. Finally, the computer or machine solves the problem by interacting with the reality to execute the problem-solving process in the physical situation realm. Hence, the generalization principle (Table 1) of scientific study, applied to problem solving activity is upheld by Approach 2. In the remaining part of this section, we will focus on the sense that computer science is science is a problem-solving activity rather than human behaviour simulation. This sense of computer science is science has direct linkage with the engineering science theory which is about problem-solving with a man-made device put into a scientific framework.

Apart from generalization, the quality of the scientific knowledge in the aim of scientific study is also important. One aspect of the quality is the accuracy of the scientific knowledge. In the case of computer science interpreted as a problem-solving activity as in Approach 2, the success identification principle in the engineering science theory is inherited by the computer science theory. According to this principle, it predicts that the cost-effectiveness of the problem-solving activity is 100%. This is similar to the Probability Ranking Principle in information retrieval (Luk, 2020) which predicts that the accuracy is optimal. As the machines are the most powerful, the only thing that can change in the problem-solving activity is the algorithm or the program. Therefore, the prediction of the cost-effectiveness applies to the algorithm. The cost-effectiveness is measured by two aspects: one on how effective, E, is the problem solved, which we can give a percentage, and the (normalized) cost, NC, which is another percentage. Thus, one cost-effectiveness measure, CE, is E – NC which can range from 1 to -1. For algorithms that guarantees to solve the problem, E is 100%, as for many combinatorial problems. However, the algorithms compete with each other by having the smallest NC so that the overall CE is the largest. That is why some computer scientists work on papers

that focuses on the computational complexity (i.e., the cost) and has the proof that the problem is solved (i.e., the effectiveness). The prediction accuracy of the problem-solving activity is the prediction accuracy of the cost-effectiveness. This would encourage algorithms with higher CEs so that the prediction error of the problem-solving ability is less, because the error is 100% - CE achieved. In this way, the more cost-effective the algorithm is the more accurate the cost-effectiveness prediction and so the knowledge in the algorithm is better in a scientific sense as accuracy is a quality of the scientific knowledge. Note that since we do not know beforehand whether the E measure of the algorithm will be 100%, the success identification principle is testable, and therefore the principle of empiricism (Table 1) in scientific study is upheld. Also, the modelling accuracy is based on randomly guessing the answer to a problem of the solution. Therefore, we expect that most computer solutions or applications have better Es than the E by random guess. We will discuss this in more details in Sect. 6. In summary, we expect that the modelling accuracy principle (Table 1) is upheld.

Another quality of scientific knowledge is reliability. For some problems, we have a proof for solving them based on the algorithm so that E is 100% and this can be absolutely reliable. However, the measure of the accuracy of the scientific knowledge in computer science is based on cost-effectiveness. Therefore, we need to consider whether the prediction of NC is reliable or not. This means that we have to deal with computational complexity (CC). CC is typically measured in terms of time-cost and space-cost. Moreover, computer scientists are usually not concerned with the actual time-cost and space-cost, which may depend on a number factors (like what CPUs are used) but on the (upper) bounds of the time-cost and space-cost. The reason is that if the computer scientists feel that the bounds of the time-cost and space-cost are small enough for the problem size that they are interested in, then they think the solution is good enough. Therefore, NC is measured in terms of the (upper) bounds of costs for the specific problem sizes that the scientists are interested in. Because NC is dependent on the problem size, so is E dependent on the problem size. In this case, the reliability of E may be specified by some bounds on E rather than some exact figure of E.

The final aspect of the quality of (scientific) knowledge is consistency. According to this approach, it is about the scientific knowledge of the problem-solving activity that needs to be consistent. Because central processing units (CPUs) are very complicated circuitries, it is not unheard of that there are bugs or inconsistencies in the circuitries (e.g., Pentium FDIV bug). So, it is possible that the CPU circuit model that generates the physical circuitry may have inconsistencies. For complex programs, they may also have (semantic) bugs (Nuseibeh, 1996) in problem-solving that need to be fixed. Detecting these bugs may be very difficult because the tools (e.g., compilers) may not be able to pick

up these bugs. This subject is more in the domain of software engineering (as this may depend on the specification and requirements of the system) which we will not elaborate any further. In general, computer science is concerned with the consistencies of the computers and programs in order to solve the problems correctly.

The phenomenon of computer science can be considered as the state of the physical problem. The problem-solving activity starts at an initial physical state and based on actions of the physical computers the physical state was caused to change. The physical computer reads information from the physical state which is caused to be changed further by further actions of the physical computers. This cycle of physical state changes can be thought of as a casual chain reaching the final physical goal state. Therefore, the desired phenomenon that we want is for the physical computer to arrive at the physical goal state (i.e., our desired phenomenon to be achieved). Thus, the assumption of the causality of phenomenon (Table 1) is upheld.

The no magic assumption (Table 1) states that if identical or similar situation occurs, then identical or similar distributions of outcome are produced. This assumption may not be exactly followed for some types of programs (like randomized algorithms). Instead of solving the problem exactly, these types of programs may solve the problems probabilistically so that on average, the problem is solved better than solving the problem by random guess. The repeatability of the problem-solving ability may be called into question, but we believe that the distribution of the problem-solving ability outcome remains the same after repeated trials. Therefore, we believe that the no magic assumption still holds for these types of programs or algorithms.

6. Common outstanding issues

Why do we have computer science and not computing science? The reason is that computing science focuses on the algorithm (e.g., Knuth, 1968; Shapiro, 2001) to define computing science. While it is true that most computer scientists are concerned with designing algorithms and implementing high level language programs, the set of instructions of the algorithms and the high-level programming language are defined by the programmable device (i.e., computer). Now, not all instructions can be implemented in a computer, but all computer instructions can be used by the algorithms and high-level programs. Therefore, computers specify the allowable instruction sets used by the algorithms and high-level programs. Thus, computers are more fundamental than the algorithms or high-level programs. It is because of this dependency, computer science as a name is preferred over the name computing science. If computing science is used, we may be emphasizing computer science as

Approach 1 over Approach 2 where some (human) instructions may not be realisable by computers. In this case, our hierarchy framework of theory, model, experiment and physical situation is broken as the experiment may not have a realisable computer to execute the program, and this affects our claim that computer science is science.

Some computer scientists are only concerned with computational complexity of an algorithm like the time-space complexity. They seem to forget whether they are solving a problem or not, or what programmable device they use. Usually, these computer scientists are focused on a problem that can be solved. So, the solvability of the problem is 100% for the algorithm. Therefore, the computer scientists are no longer concerned about the solvability issue of the algorithm. Instead, they are concerned with how fast the problem can be solved and how little resources does the algorithm need to solve the problem. Therefore, these computer scientists appeared to be more concerned with the computational complexity of the algorithm. For some scientists, they are concerned with the computational complexity of the problem which may place some limit to the efficiency in solving the problem by whatever algorithms that can be designed. This kind of issues may be involved mathematically, and they may deserve special attention from the computer scientists. That is why their papers are focused on time-space complexity without touching on the other aspects of computer science.

One concern is that computer scientists rarely do experiments. This is, however, not true in general, as this depends on the kind of problem the computer scientists are trying to solve. For many combinatorial problems, computer scientists are mostly interested in the computational complexity because there are simple algorithms that can enumerate the solutions and guarantee to solve the problem with 100% solvability but with poor computational complexity. These computer scientists are usually not concerned with the real-time cost or exact space-cost because they are trying to solve the problem efficiently when the problem size is large, and because improvement of computer technology may mean that real-time cost or space cost may become obsolete as technology advances, so that we are only interested in the general form of the complexity (e.g., whether it is polynomial or exponential cost) rather than the exact form. Therefore, computer scientists rarely perform experiments to measure the time-cost or space cost. Having said that, some computer scientists are concerned with the real-time cost and space cost. These may be computer scientists specialized in real-time systems or control systems where real-time response is required. For those computer scientists, they may look at the real-time cost and space-cost efficiency of the algorithms. In these cases, experiments may be carried out to take actual measurements of efficiency. For other problems where the solvability of the problem cannot guarantee 100%, experiments are done to benchmark the performance of the algorithms or

models. For example, information retrieval as a sub-discipline of computer science is concerned with the accuracy of the retrieval, which is usually not 100%. Therefore, information retrieval has published many papers with experimental results (Luk, 2020). Other sub-disciplines of computer science like neural network, computer vision, pattern recognition and machine learning have many papers that report experiments on performance. Therefore, there is a large class of computer science works that involve experiments.

One concern of the modelling accuracy principle is that random guess is used to define the lower bound performance. In computer science, some algorithms (like randomized algorithms) may solve a problem by making random guess, so it may appear that some problems in computer science can never surpass the performance of random guess. It should be noted that the algorithms that make use of random guess do not just guess the solution without any knowledge or structure. Instead, these algorithms may employ certain knowledge or structure in the solution, and only in certain part, random guess is made to help solving the problem. Therefore, such algorithms should not be considered as pure random guess algorithms which are used to define the lower bound modelling accuracy performance. Thus, the algorithms in computer science involving random guess may perform better than the pure random guess algorithm used to define the lower bound modelling accuracy, so that the modelling accuracy principle (Table 1) is upheld.

Is computability still a central question in computer science? For both approaches, this question is indeed central. The reason is that this question probes the limit of computer science because it asks the questions what problems can be solved by the methodology that solves a problem by following a sequence of instructions. This limit depends on whether the instruction is implementable in a machine, which set some limits for Approach 1. This limit also depends on whether the most powerful computer ever devised cannot solve what problem, which sets the limit of Approach 2.

7. Conclusion

In this paper, we have defined computer science as a problem-solving activity involving a programmable device (model). We explain why the field is called computer science and not computing science because all instructions in the algorithm are realizable by a computer (or a programmable device). We have also highlighted how computer science correspond to science based on two approaches. The first approach considers computer science as simulation of human behaviour which is similar to the goal of artificial intelligence. However, this approach decouples the simulation accuracy from the cost-effectiveness of problem solving as a result this decouples the approach from some of the

principles of computer science and scientific study in general, so this approach is less preferred. The second approach considers computer science based on the theory of computation. It guarantees that the simulation accuracy of human behaviour is 100% for Turing computers since the instructions can be carried out by a human being. Based on the theory of computation, computer science guarantees that the algorithms can be executed in a programmable device that can be realized. Using the programmable device, algorithms are defined as sequences of high-level instructions that are realizable in programmable devices. This definition makes clear the vague notion of an algorithm before.

One major insight in this paper is that computer science is about using a programmable device to solve technical problems. While it is not a surprise that a programmable device is used, what is surprising is that computer science is regarded as problem solving. This is different from past influential definitions of computer science which is related to algorithms or information. Regarding computer science as problem solving enables computer science to be related to engineering science, as well as enabling the success identification principle to be directly applied to the problem-solving instances so that there is a prediction of the cost-effectiveness of the solution. In turn, this prediction is related to the accuracy of the scientific knowledge which consists of the problem model and the algorithm. As a result, this problem-solving perspective of computer science unifies the various aspects of computing into a framework of theory, model, experiment and physical situation, enabling us to claim computer science is science. This insight also concurs with the informal definition of computer science as "solving problems with the aid of a computer" (Margolis et al., 2008).

Another major insight in this paper is that computer science can be regarded as simulation of human behaviour for both Approaches 1 and 2. Effectively, the computer is simulating an agent who follows the given instructions to solve a problem. This simulation can be made to relate to artificial intelligence, where complex instructions are allowed to specify in the procedure to solve the problem. In the case that the instructions are limited to those executable by a von Neumann machine, the simulation of the agent has an accuracy of 100% since the instructions of the von Neumann machine are simple ones that a human and a machine can both follow without difficulty. In this sense, Approach 2 is doubly scientific as the computer simulates human doing the problem solving, and the problem-solving activity is a scientific one.

The final major insight in this paper is that there is a scientific discipline called engineering science. It can be divided into applied science sub-disciplines (like mechanical engineering science) and pure science sub-disciplines, in which computer science is an example. Engineering science has its own assumptions and principles. One can even develop a framework of theory, model, experiment and physical situation for engineering science. Specifically, the theory contains the aim, assumptions and principles mentioned in

engineering science plus the problem type and the man-made device. In the scientific model realm, the man-made device is specialized into a device model and the problem type is specialized into a problem model. The device model is then realized as a physical device in the experiment realm and the problem model is specialized into a problem instance group, which is manifested in the physical problem situation or physical state in the physical situation realm. The device will change this physical problem state into a physical goal state that solved the problem in the physical situation realm. This is very similar to the framework (Figure 2) mentioned for Approach 2 in computer science as this is a problem-solving activity in engineering science and in computer science. The minor difference between computer science and engineering science in general is that computer science specify that a von Neumann/Turing machine is used, but the engineering science may be a von Neumann/Turing machine or some other device that the engineering scientist creates, as well as the freedom of not requiring the Church-Turing thesis to justify the use of the Turing machine in engineering science theory so there is no necessity to develop an algorithm/program to solve problems.

References

[1] Anguera, A., Lara, J.A., Lizcano, D., Martinez, M-A., Pazos, J. and David de la Pena, F. (2020) Turing: the great unknown. *Foundations of Science*, 25(4), 1203-1225.

[2] Balsamo, B.H., Francois, C., Kauffman, L., Klir, G., Mandel, T., Rhee, P.Y., Sabelli, H. and Salthe, S. (2000) What are the principles of system science? In *Roundtable of the World Congress of the System Sciences*, Toronto.

[3] Boon, M. (2008) Diagrammatic models in the engineering sciences. *Foundations of Science*, 13(2), 127-142.

[4] Denning, P.J. (2003) Great principles of computing. *Communications of the ACM*, 46(11), 15-20.

[5] Denning, P.J. (2005) Is computer science a science? *Communications of the ACM*, 48(4), 27-31.

[6] Denning, P.J. (2007) Computing is a natural science. *Communications of the ACM*, 50(7), 13-18.

[7] Dershowitz, N. and Gurevich, Y. (2008) A natural axiomatization of computability and proof of Church's thesis. *The Bulletin of Symbolic Logic*, 14(3), 299-350.

[8] Dlouhy, J. (2011) *Richard Feynman on computer science – talk at Bell labs (1985)*. http://youtube.com/watch?v=IL4wg6ZAFIM [accessed: July 2, 2020].

[9] Eden, A.H. (2007) Three paradigms of computer science. *Minds & Machines*, 17(2), 135-167.

[10] Fuhr, N. 2012. Salton award lecture information retrieval as an engineering science. *ACM SIGIR Forum*, 46(2), 19.

[11] Gurevich, Y. (2000) Sequential abstract state machines capture sequential algorithms. *ACM Transactions on Computational Logic*, 1(1), 77-111.

[12] Klir, G. and Wierman, M. (1999) *Uncertainty-based information: elements of generalized information theory*. Heidelberg, Germany: Physica-Verlag.

[13] Knuth, D.E. (1974) Computer science and its relation to mathematics. *The American Mathematical Monthly*, 81(4), 323-343.

[14] Knuth, D.E. (1968) *The art of computer programming, Vol I: fundamental algorithms*. Addison-Wesley.

[15] Krebsbach, K.D. (2015) Computer science: not about computers, not science. In *Proceedings of the 2015 International Conference on Frontiers in Education: Computer Science and Computer Engineering*, CSREA Press, Las Vegas, Nevada.

[16] Krueger, T., Panknin, D., and Braun, M. (2015) Fast cross-validation via sequential testing. *Journal of Machine Learning Research*, 16(33), 1103-1155.

[17] Luk, R.W.P. (2010) Understanding scientific study via process modelling. *Foundations of Science*, 15(1), 49-78.

[18] Luk, R.W.P. (2017) A theory of scientific study. *Foundations of Science*, 22(1), 11-38.

[19] Luk, R.W.P (2018) On the implications and extensions of Luk's theory and model of scientific study. *Foundations of Science*, 23(1), 103-118.

[20] Luk, R.W.P. (2020) Why is information retrieval a scientific discipline? *Foundations of Science*. Doi: 10.1007/s10699-020-09685-x.

[21] Margolis, J., Estrella, R., Goode, J., Holme, J.J. and Nao, K. (2008) *Stuck in the shallow end: education, race and computing*. Cambridge, Massachusetts: MIT press.

[22] McCarthy, J. (1962) Towards a mathematical science of computation. *Proceedings of IFIP, Congress 62*.

[23] Miller, B. and Ranum, D. (2013) *Problem solving with algorithms and data structures*.

http://www.cs.auckland.ac.nz/compsci105s1c/resources/ProblemSolvingwithAlgorithmsandDataStructures.pdf [accessed: July 23, 2020].

[24] MIT OpenCourseWare (2009) *Lecture 1A | MIT 6.001 structure and interpretation, 1986*. http://youtube.com/watch?v=2Op3QLzMgSY [accessed: July 2, 2020].

[25] Moore, J.S. (2014) Proof pearl: proving a simple Von Neumann machine is Turing complete. In *Proceedings of the International Conference on Interactive Theorem Proving* (pp. 406-420).

[26] Newell, A., Perlis, A.J. and Simon, H.A. (1967) Computer science. *Science*, 157(3795), 1373-1374.

[27] Newell, A. and Simon, H.A. 1976. Computer science as empirical inquiry: symbols and search. *Communications of the ACM*, 19(3), 113-126.

[28] Nuseibeh, B. (1996) To be and not to be: on managing inconsistencies in software development. In *Proceedings of the 8th IEEE International Workshop on Software Specification & Design* (pp. 164-169).

[29] Polak, P. (2016) Computing as empirical science – evolution of a concept. *Studies in Logic, Grammar and Rhetoric*, 48(61), 49-69.

[30] Rapaport, W.J. (2017) What is computer science? *APA Newsletter on Philosophy and Computers*, 16(2), 1-22.

[31] Rapaport, W.J. (2018) What is a computer? A survey. *Minds & Machines*, 28(3), 385-426.

[32] Rapaport, W.J. (2020) *Philosophy of computer science*. http://cse.buffalo.edu/~rapaport/Papers/phics.pdf [accessed: July 3, 2020].

[33] Shapiro, S.S. (2001) *Computer science: the study of procedures*. http://www.cse.buffalo.edu/~shapiro/Papers/whatiscs.pdf [accessed: July 3, 2020].

[34] Shannon, C. (1948) A mathematical theory of communication. *Bell System Technical Journal*, 27(3), 379-423.

[35] Simon, H.A. (1969) *The sciences of the artificial*. Boston: MIT Press.

[36] Wegner, P. (1968) *Programming languages, information structures and machine organization*. McGraw-Hill.

[37] Wegner, P. (1976) Research paradigms in computer science. In *Proceedings of the 2nd International Conference on Software Engineering*, San Francisco, C.A. (pp. 322-330)

[38] Weinburg, G.M. (2001) *An introduction to general systems thinking*. New York: Dorset House.