Modelling interactive computing systems: Do we have a good theory of what computers are?

Alice Martin, Mathieu Magnaudet, Stéphane Conversy

Modelling interactive computing systems: Do we have a good theory of what computers are?

Alice Martin
Mathieu Magnaudet
Stéphane Conversy
ENAC, Université de Toulouse

Abstract

Computers are increasingly interactive. They are no more transformational systems producing a final output after a finite execution. Instead, they continuously react in time to external events that modify the course of computing execution. While philosophers have been interested in conceptualizing computers for a long time, they seem to have paid little attention to the specificities of interactive computing. We propose to tackle this issue by surveying the literature in theoretical computer science, where one can find explicit proposals for a model of interactive computing. In that field, the formal modelling of interactive computing systems has been brought down to whether the new interaction models are reducible to Turing Machines. There are three areas where interaction models are framed. The comparison between TMs and interactive system models is at stake in all of them. These areas are namely some works on concurrency by Milner, on Reactive Turing Machines, and on interaction as a new computing paradigm. For each of the three identified models, we present its motivation, sum up its account for interaction and its legacy, and point out issues regarding the understanding of computers. The survey shows difficulties for epistemologists. The reason is that these analyses focus

Philosophical Problems in Science (Zagadnienia Filozoficzne w Nauce)
No 73 (2022), pp. 77–119

on the formal equivalence between interactive models of computation and classic ones. Such a project is different from addressing how a computing machine can be interactive: in other words, which mechanisms allow it.

Keywords

philosophy of computing, models of computation, interactive computing, computing mechanism, computational mechanistic explanation.

Introduction

In the philosophy of computing, we are paying increased attention to a set of new features of computers. This set has led to the introduction of a new label for these computing machines: they are referred to as interactive computing machines (Dodig-Crnkovic, 2011; Goldin, Wegner and Smolka, 2006; Soare, 2013; Van Leeuwen and Wiedermann, 2001; Wegner, 1997). The set of new features can be captured in the following statement made in a 2011 paper by Gordana Dodig-Crnkovic (our highlights):

> Present day computers are very different from the early standalone calculators designed for mechanizing mathematical operations. They are largely *used for communication in world-wide networks* and variety of information processing and knowledge management. Moreover, they play *an important role in the control of physical processes and thus connect to the physical world*, especially in automation and robotics. [...] Computational processes are nowadays *distributed, reactive, agent-based, and concurrent*. The main criterion of success of the computation is not its termination, but its *response to the*

> *outside world*, its speed, generality and flexibility; adaptability, and tolerance to noise, error, faults, and damage (Dodig-Crnkovic, 2011).

Historically, the concept of interaction was introduced by a computer scientist, Milner, in the 1970s-1980s (Milner, 1975; 1982; 1993; 1999). At first, an *interactive computing system* was defined as a system where several threads execute instructions in parallel while being able to synchronize and communicate at certain moments of the execution. Since then, the characteristics of computer systems have continued to evolve, and by "interactive" we refer today to a broader set of properties that can be grouped as follows: the ability to continuously react in time to external events that modify computing execution. This class of computers deserves all our attention since they are ubiquitous. Every computer system today is designed to respond to external events in a predictable way and according to temporal constraints. In any case, what distinguishes this class of so-called interactive computing machines from the classical computer systems that preceded them is that they are no longer purely *transformational systems*. A *transformational system* is a classical computing device that, given a set of inputs, produces a final output after a finite execution. This evolution of computing complicates the answer to *what a computer is*. The question is well-known in the philosophy of computing (Piccinini, 2008; Rapaport, 2018; Smith, 2002). As already noted, many answers to the question distort it and answer the question of *what a computation is*, immediately projecting the field of investigation into the theory of computability:

> A fairly obvious, trivial, and almost-circular definition of 'computer' says that a computer is a machine that computes. The natural next question is: What does it mean to compute? But this shifts the burden of answering our question away from

> what computers are to the topic of what computation is. Many of the objections to various theories about computers are really objections to what counts as a computation (Rapaport, 2018).

This leaves us with the specific issue we want to address. We ask whether models of computation for interaction allow us to answer the question of what an actual (necessarily interactive) computer is. Current computers come in various forms and we chose in this paper to restrict our concerns to a delimited notion of *interaction*, as defined in Human-Computer Interaction (Basman et al., 2018; Beaudouin-Lafon, 2006; Dearden and Harrison, 1997; Hornbaek and Oulasvirta, 2017; Myers, 1994), and target a specific set of ubiquitous computing devices—those interacting with humans, e.g., through digital interfaces. We will not elaborate on *analog computing* (Bielecki, 2019) and *natural computation* (Dodig-Crnkovic, 2011; MacLennan, 2003).

To tackle the issue of interactive computing devices, we propose here an approach that, to the best of our knowledge, has not been proposed so far: we want to examine the models of computation proposed in theoretical computer science to think about interactive computing systems. We offer a literature survey where one can find *explicit theories of interaction*.[1] We show that the formal modelling of interactive computing systems has been brought down to whether the new interaction models are reducible to Turing's *a-machines* (Turing, 1937)—we will refer to them as Turing Machines (TMs). Questioning the theoretical bounds of the Turing Machine in computer science when faced with the existence of interactive computing devices has been explored at least since Milner's work on communicating and

[1] We insist on our two criteria: *explicit theories of interaction* in *theoretical computer science*. We have in mind the fact that other communities e.g., the engineering community on reactive systems, are related to our topic but they have not conceptualized *interaction* as such.

mobile systems (Milner, 1993; 1999). To the best of our knowledge, there are three areas where interaction models are framed as such. These areas are some works (i) on concurrency by Milner and his followers (Milner, 1999; 2006), (ii) on Reactive Turing Machines (Andersen, Mørk and Sørensen, 1997; Baeten, Luttik and Tilburg, 2013; Van Leeuwen and Wiedermann, 2001; 2006), and (iii) on interaction as a new non-algorithmic computing paradigm (Goldin, Wegner and Smolka, 2006; Wegner, 1997; Wegner and Goldin, 2003). For each of the three identified models, we:

- present the motivation behind it,
- sum up its account for interaction,
- identify its legacy,
- point out issues regarding the understanding of computers qua that model.

We then want to show how these approaches, which belong to a formal approach, cannot provide an answer to the question of what computers are, and for two reasons. On the one hand, these models of computation have focused their attention on whether interactive models are reducible to models of classical computation—par excellence, the Turing machine. Proving (or not) that an interactive property can be formalized as a computable property in the classical Turing's sense does not answer the question of how an interactive property comes into existence and can be the object of execution. On the other hand, and this is a correlate, these models do not propose a basis for a mechanistic explanation of the very possibility of an interactive computing system. With only formal models of interactive computation, we might run the risk of not offering an adequate conceptualization

of current computers. Therefore, we end up proposing to take the distinction seriously between *models of computation* and *mechanistic computational explanations*, as presented by Miłkowski (2011; 2014).

1. Milner introduces a distinction between interactional and computational behavior

1.1 Motivation

Milner was the one who introduced the concept of interaction in computer science. He summarized his motivations in a famous Turing Award speech (Milner, 1993). Milner was concerned with the logical foundations of computing inherited from Turing. He was preoccupied with the idea that computing practices had evolved since the birth of computing, notably in terms of architecture. He took seriously the possibility that the logical foundations dating back to the thirties may not match the growing challenges of his time and may require additional concepts.

Milner (2006) pointed out that the logical foundations of computing offered by Turing (1937) were previous to the first physical computers and that computer science is grounded in *logic* and *engineering*. On the engineering side, computer science was inherited from von Neumann's pioneering work (Aspray, 1990; Godfrey and Hendry, 1993). Only one thing could happen at once in an early von Neumann's computer. Nevertheless, there was more to computing than von Neumann's architecture (Backus, 1978; Milner, 2006). A growing interest in dealing with concurrency in the sixties and seventies made sequential programming less warranted. Therefore, to Milner, the logical foundations of computing were to evolve. The main flaw of the early logical foundations was the reduction of computing processes

to the concept of an algorithm, which tends to associate computing with mere calculation without taking concurrent activity into account. Because of the evolution of computing engineering practice, Milner questioned whether the logical grounds of computing should evolve as well. Milner's thesis can be put in a nutshell: "this logical foundation has changed a lot since Turing but harks back to him. To be more precise: (i) Computing has grown into informatics—the science of interactive systems; (ii) Thesis: Turing's logical computing machines are matched by a logic of interaction" (Milner, 2006). Consequently, a theory and new language to express concurrent activity were required: "we must find an elementary model which does for interaction what Turing's logical machines do for computation" (Milner, 2006).

The need to define a new computing theory is first displayed through the evolution of computing practice. To sum up, Milner's motivation and focus were the solving of concurrency issues in distributed systems, with the idea that the evolution of computing practices required new formal tools: "Through the 1970s, I became convinced that a theory of concurrency and interaction requires a new conceptual framework, not just a refinement of what we find natural for sequential [algorithmic] computing" (Milner, 1993).

1.2 Account for interaction

Milner introduced the opposition between *interactional* and *computational* behaviour. Introducing the concept of interaction, Milner (1975; 1982; 1983) referred to concurrent message passing between agents. Milner's work coincided with Petri's (1980) new model of concurrent processes, which generally intended to describe concurrency in information systems.[2] To Milner, interaction is more *expressive* than a TM,

[2] Concurrency theory emerged from Dick Karp's early work in the 1960s, grew with (Petri, 1980) and later work on transition systems (Glabbeek and Plotkin, 2004; Nielsen,

but it still describes an *effective* procedure. Milner did not assert *equivalence* between an interactive model and a TM, but he introduced the topic (Milner, 1999) and left it unanswered. Four main differences between old (computational) and new (interactional) computing are made striking by Milner. First, in Milner's words, a Turing Machine prescribes a behaviour to be executed. In contrast, new computing requires the description of an information flow between several system components. Second, old computing is characterized by a *hierarchical design* when current practice involves *heterarchical* phenomena in the computing system. Third, in new computing, the designer cannot predict when agents will be triggered or the overall behaviour of the computing system. Fourth, the user is not merely looking for an end result in new computing practice. There is more than a mathematical function to evaluate, as it used to be in old computing. The user instead interacts with the system, and the look for an end result is replaced by continuing interaction. Having taken stock of the evolution of computing practice on the engineering side, Milner examines its consequences on the logic foundations of computing. The pi-calculus and his work on the equivalence with automata, known as bisimulation, achieved this reflection on interactive processes (Milner, 1993; 1999) with a formalism.

1.3 Legacy

Milner's work on interaction has become a founding block in automata theory and concurrency theory. It installed the notion of a transition system as the prime mathematical model to represent discrete behavior (Arbach et al., 2015; Baldan, Corradini and Montanari, 2001;

Plotkin and Winskel, 1981), and has now developed into a mature theory of reactive systems (Harel and Pnueli, 1985) with diverse network models (for an overview, see Lee and Sangiovanni-Vincentelli, 1998; Lee and Neuendorffer, 2006).

Glabbeek and Plotkin, 2004; Nielsen, Plotkin and Winskel, 1981). It also showed that language equivalence was not the correct notion when comparing automata for interactive systems. Instead, it should be replaced by a notion of behavioral equivalence or bisimilarity (Milner, 1999). The pi-calculus has inspired research to derive a language from it. The *Pict* (Pierce and Turner, 2000) programming language is an example.

Milner's work is foundational and served as a reference for anyone after him, reflecting on the need for a new framework dedicated to new emerging computing practices. Milner insists on an essential reminder that we would like to consider. When modelling, the engineering practice matters and is to be articulated with the logical foundations of the model, should it involve elaborating a new framework. Famously, Wegner and Goldin acknowledge that Milner was the first to introduce the idea that classic models of computation were insufficient. They argue that Milner did not state clearly whether computation in concurrent communicating systems (CCS) and the pi-calculus were reducible to Turing machines and algorithms (Wegner and Goldin, 2003). If one goes and looks at Milner's Turing Award Speech, it seems true that classical computation translates into an interactive calculus. However, it is not stated whether any formula in the pi-calculus can be expressed in a classical calculus like the lambda-calculus.

1.4 Issues for an account of current computers

Given the account of current computers that we are looking for, we see two limits in the lessons drawn from Milner. First, we are looking for an explanation of the interactive computing phenomena at stake in a computer. Therefore, the relation between layers of abstraction, from

the computational to the physical, is crucial. However, to Milner, the physical layer of the machine is not of much interest, and the calculus of CCS needs to abstract away from the physical. As Milner puts it, informatics is about virtual symbols: "physical systems tend to have permanent physical links; they have fixed structure. But most systems in the informatic world are not physical; their links may be virtual or symbolic" (Milner, 1999). From our perspective, abstracting away from the physical world comes at some cost for an explanation. A complete explanation of a computing system can hardly be provided in details within a single understandable abstraction, since a computing sytems is extremely multi-layered (Lee, 2020; Nisan and Schocken, 2005). Therefore, an explanation of a computing system is necessarily a trade-off between understandability and overwhelming details. As we will flesh out in the last section 4 by referring to Miłkowski's work (Miłkowski, 2011; 2016), a good computational explanation must link the formal story and the blueprint of the computing mechanism. Such articulation is not told in a formal theory of concurrent processes. Second, this first story of interactive systems restricts them to concurrent systems, which is only one dimension of interest when describing what current computers do. There are *at least* two core dimensions left aside: what makes possible timing instructions and the connection between physical processes inside and outside the computing system.

2. Reactive TMs: extending the original model

2.1 Motivation

More recently, a literature domain focused on a "Reactive Turing machine" has emerged (Andersen, Mørk and Sørensen, 1997; Baeten, Luttik and Tilburg, 2012; 2013; Luttik and Yang, 2016; Van Leeuwen

and Wiedermann, 2006). It reminds us that the purpose of Turing's *a-machine* model was to propose a formal account of what is *computable by effective means* (algorithmically computable). This formalization was achieved before the realization of the first digital computers. In a way reminiscent of Milner, the question is whether the TM model still fits computing practices decades later. The strategy chosen is to see whether *extensions* of the original TM are sufficient to describe new computing practices and whether the obtained model is still equivalent to a TM. The strategy founds its frame within computability theory and reflects on its scope. This literature domain that proposes extensions of the Turing machine to account for interactive computing systems may be traced back to seminal works on a "Universal reactive machine" (Andersen, Mørk and Sørensen, 1997). In that respect, although pointing at the specificity of interactional behaviour, the main framework still relates to Turing's. Baeten, Luttik, and van Tilburg (Baeten, Luttik and Tilburg, 2013) are looking for a model of interactive computation, extending the classical TM with a process-theoretical notion of interaction related to Milner's previous work. The strategy involves questioning the relationship between such extensions and the Church-Turing thesis. As a reminder, the Church-Turing thesis states that a computable function by effective means is computable by a Turing machine. The community interested in Reactive Turing machines asks the following question: can the Church-Turing thesis also be extended? Van Leeuwen and Wiedermann (2001) focus on the possible extension of the Church-Turing thesis to account for interactive computing: "We will motivate the need for a reconsideration of the classical Turing machine paradigm and formulate an extension of the Church-Turing thesis" (Van Leeuwen and Wiedermann, 2001).

What is at stake is whether the Church-Turing thesis holds given warranted new models of computation: "Is the Church-Turing thesis

as we know it still applicable to the novel ways in which computers are now used in modern information technology? Will it hold for the emerging computing systems of the future?" (Van Leeuwen and Wiedermann, 2001). The Church-Turing thesis originally did not entail a claim about computing in general (what computers do and will do) but only about *effective computation*. Therefore, it does not follow that we should ask the Church-Turing thesis for answers on what computing is. Replacing a question about computing with a question about computation is the mark of a specific formal perspective within the frame of computability theory. Understanding *computing* and its evolution from a formal perspective consist of questioning *what can be computed* and seeing if there is another notion of computation than effective computation *in the sense of Church-Turing*.

2.2 Account for interaction

The starting point in the Reactive TM community is a standard current computer designed as a distributed system interacting with an environmental agent: a *site machine*. Starting from this model, the reflection on interaction aims at showing the equivalence between this site machine and a Turing machine augmented by some functions. The conclusion is that a site machine computer computes effectively and yet requires a TM with new functions, thus requiring an extension of Church-Turing's thesis. There are effectively computable functions that TMs, in a strict sense, cannot compute. One crucial dimension that the community wants to account for is particularly relevant to us: "In order to mimic site machines, a Turing machine must have a mechanism that will enable it to model the change of hardware or software by an operating agent" (Van Leeuwen and Wiedermann, 2001). To make interaction with an external agent possible, the model

needs to integrate a way of entering new, external, and possibly non-computable information into the machine. This is precisely what oracles do. The authors prefer a more general notion: an *advice function*. The model of a Reactive TM (also called a TM with advice) is considered expressive and definitionally equivalent to an Oracle Turing Machine.

Van Leeuwen and Wiedermann identify three key elements that should be integrated all together within the frame of algorithmic computability: "non-uniformity of programs", "interaction of machines", and "infinity of operation". By the non-uniformity of programs, the authors refer to the fact that current programs on a personal computer are no longer fixed but evolve, are upgraded, and their data remain in memory even when the machine is not running. By interaction, they intend to contrast a TM, where all input data are present before the start of the computing procedure, with a modern computer, where continuous streaming of data via input ports is going on. The third mentioned characteristic, the infinity of operation, refers to distributed and mobile communication systems. These systems are to be seen as dynamic networks of many entities sending and receiving signals in unpredictable ways that are to be synchronized. To accommodate the original TM model, Leeuwen and Wiedermann propose to define "Interactive Turing machines with advice." Integrating an "advice" function amounts to entering new, external, and non-computable information into the machine, which requires using oracles (Balcázar, Díaz and Gabarró, 1995; Rogers, 1987). This way, a TM with advice resembles site machines and I/O automata in being equipped with input and output ports. To the authors, formal tools to support interaction and infinite computations are already available. As for interaction, they refer to already well-known and developed literature on the theory of concurrent processes, the programming of parallel processes,

communication protocols, and distributed algorithms. As for infinite computations, Leeuwen and Wiedermann understand them from the language-theoretic viewpoint in the theory of omega-automata (Staiger, 1997; Thomas, 1990).

2.3 Legacy

This approach to extending the Turing machine and the Church-Turing thesis is at the junction between Milner's work and Wegner's (presented in the coming section 3). It makes the junction in that it begs the question of a new paradigm. Milner had not formulated his theory of interaction in such radical terms, but Wegner goes further. The Reactive Turing Machine community asks whether the mentioned required extensions lead to a new computing paradigm: "The experience with present-day computing confronts us with phenomena that are not captured in the scenario of classical Turing machines" (Van Leeuwen and Wiedermann, 2001). The computations carried out on Turing machines with advice are said to be "more powerful" than classic computations on a-machines. The authors insist that this claim does not go against the Church-Turing thesis. To Leeuwen and Wiedermann, like other physical systems (Pour-El, 1999), TMs with advice or oracle Turing machines do not fit the concept of a finite algorithm that can be computed by means of a TM. The conclusion pushes towards a paradigm shift:

> What makes them non-fitting under the traditional notion of algorithms is their potentially endless evolution in time. This includes both interaction and non-uniformity aspects. This gives them the necessary infinite non-uniform dimension that boosts their computational power beyond that of standard Turing machines (Van Leeuwen and Wiedermann, 2001).

The authors ensure that such a paradigm shift does not put into question the original Church-Turing thesis because their proposal for interactive computation does not involve solving undecidable problems (Van Leeuwen and Wiedermann, 2001) using effective computation. The work seems to have served as a pivotal point in structuring the debate on a model of interactive computation around its implications for the Church-Turing thesis. This is evidenced by the objections formulated against Wegner's work which pushes further the concept of interaction and the need for a new paradigm: a proposal of this kind had fallen under objections framed within the theory of computability.

2.4 Issues for an account of current computers

The project is focused on extending the original TM to make it "reactive". The proposed level of abstraction cannot account for the mechanisms that make the proposed extensions possible. We can take a closer look at the type of description presented in this formal framework to account for an interactive scenario:

> The computational scenario of an interactive Turing machine is as follows. The machine starts its computation with empty tapes. It is driven by a standard Turing machine program. At each step, the machine reads the symbols appearing at its input ports. At the same time, it writes some symbols to its output ports. Based on the current context, i.e., on the symbols read on the input ports and in the 'window' on its tapes, and on the current state, the machine prints new symbols under its heads, moves its windows by one cell to the left or to the right or leaves them as they are, and enters a new state. Assuming there is a move for every situation (context) encountered by the machine, the machine will operate in this manner forever. Doing so, its memory (i.e., the amount of rewritten tape) can

> grow beyond any limit. At any time $t > 0$, we will also allow the machine to consult its advice, but only for values of at most t (Van Leeuwen and Wiedermann, 2001).

If we look for a mechanistic explanation of computing, we need some elements to be unpacked beyond a formal account to make sense of the quoted scenario above. For example, we need to account for how reading and writing on the ports are possible. It presupposes that the interactive computing system can wait, pause, and react depending on the arrival or absence of new data. What allows such behavior? It presupposes some mechanisms allowing the system either to be interrupted by environmental processes or to check the new incoming values steadily.[3] In other words, given the initial question ("what is an interactive computer?"), some phenomena cannot be accounted for within the frame of an extended Turing machine. The way oracles work remains at a level of abstraction too remote from the minimal causal blueprint we need for our purpose.

3. Going beyond TMs? Wegner's new paradigm

3.1 Motivation

A strong motivation for Wegner's view on interaction is to overcome the Strong Church-Turing thesis (CTT) that he takes to prevent us from fully admitting a new paradigm in computer science. A paper fleshes out in detail clarifications against the CTT:

> The classical view of computing positions computation as a closed-box transformation of inputs (rational numbers or

[3] More on these mechanisms and on the limitations of oracles can be found in (Martin, Magnaudet and Conversy, forthcoming).

> finite strings) to outputs. According to the interactive view of computing, computation is an ongoing interactive process rather than a function-based transformation of an input to an output. Specifically, communication with the outside world happens during the computation, not before or after it. This approach radically changes our understanding of what computation is and how it is modelled. The acceptance of interaction as a new paradigm is hindered by the Strong Church-Turing Thesis (SCT), the widespread belief that Turing Machines (TMs) capture all computation, so models of computation more expressive than TMs are impossible (Goldin and Wegner, 2008).

In other words, the strong CTT stipulates that a TM could solve all computational problems and could compute anything that any computer can compute. Wegner argues that Turing himself would have denied it, referring to Turing's famous paper (Turing, 1937), as he did not only introduce TMs (calling them automatic machines, or a-machines) but did also introduce choice machines (*c-machines*), extending TMs by allowing a human operator to make choices during the computation. Turing did not view *c-machines* as reducible to TMs, suggesting other forms of computation might exist. Goldin and Wegner also like to remind us that the CTT applies only to the computation of functions rather than to all computations:

> Function-based computation transforms a finite input into a finite output in a finite amount of time, in a closed-box fashion. By contrast, the general notion of computation includes arbitrary procedures and processes—which may be open, nonterminating, and involving multiple inputs interleaved with outputs (Goldin and Wegner, 2008).

For the sake of clarity, Goldin and Wegner propose to formulate the assumptions of the CTT in their proper formulation free of extrapolation (Goldin and Wegner, 2008) explicitly:

- i. "All algorithmic problems are function-based."
- ii. "All function-based problems can be described by an algorithm."
- iii. "Algorithms are what early computers used to do."
- iv. "TMs serve as a general model for early computers."
- v. "TMs can simulate any algorithmic computing device."
- vi. "TMs cannot compute all problems, nor can they do everything that real computers can do."

One reason the strong CTT is "impossible" (Eberbach, Goldin and Wegner, 2004) is that no computable function would determine, given some finite amount of a priori information, all the real-world factors that are necessary to ensure the safe arrival of a car at its destination. An assertion to the contrary would endow TMs with the power to predict the future. Therefore, Wegner introduced *interaction* as a new paradigm, based on an empiricist approach (Wegner, 1995), to broaden algorithmic problem-solving. The reason is that Wegner and his followers take computing machines to be about physical processes, chaotic in nature (Siegelmann, 1995), requiring demanding precision to be controlled (Hartmanis, 1994). Superposed layers of abstractions allow us to describe and control those physical and chaotic computing machines. The challenge is then to bridge the gap between all those layers of abstraction, starting with the lowest physical level. A typical problem we want to solve with computers but not computable in the classic sense would be, e.g., the problem of driving home:

> the problem of driving home from work is computable—by a control mechanism, as in a robotic car, that continuously

> receives video input of the road and actuates the wheel and brakes accordingly. This computation, just as that of operating systems, is interactive, where input and output happen during the computation, not before or after it (Goldin and Wegner, 2008).

Goldin and Wegner argue that such a notion of computation does find its counterpart neither in the theory of computation nor in the concurrency theory. The motivation that goes hand in hand with this discussion against the strong CTT is a reflection on algorithms and the scope of algorithmic problem-solving. Knuth has given a classic definition for algorithms: "An algorithm has zero or more inputs, i.e., quantities which are given to it initially before the algorithm begins" (Knuth, 1968). Following a recipe (Knuth, 1968), for example, does not actually involve algorithmic problem-solving. To know how to mix the ingredients properly, one needs to adapt to dynamic variables and feedback, such as humidity conditions and the progressive evolution of the texture of the paste that are not pre-given values before execution. To Wegner, that kind of feedback does not belong to the function-based mathematical worldview. The problem of driving home from work, like baking following a recipe, is also among those problems that Knuth meant to exclude from his definition.

3.2 Account for interaction

This leads us to Wegner's account for interaction:

> Computational problem solving requires open testing of assertions about engineering problems beyond closed-box mathematical function evaluation. Therefore, we have proposed interactive computing as an empiricist model that expands computational problem solving from algorithmic TM models

> and functional input-output to broader concepts of interleaved dynamic streams and observable interaction with the environment (Wegner and Goldin, 2006).

In Wegner's perspective, interactions are more powerful than TMs with finite initial inputs. TMs with oracles and unbounded (dynamically extensible) input streams model more accurately interactive systems than traditional Turing machines. Interactive systems react dynamically to external events. They are also related to the passage of external time. By delaying the binding time of inputs so that they can occur during the computation (rather than only at the beginning) and modelling reactive processes (Manna and Pnueli, 1992) by infinite computations (Thomas, 1990), the modelled entities are extended from algorithms to persistent objects and concurrent processes (Milner, 1999).

Wegner wonders if Milner himself avoided questioning whether the computation in CCS and the pi-calculus went beyond Turing machines and algorithms (Wegner and Goldin, 2003). The question could remain whether Wegner takes interaction as a super-calculus/super-algorithm or as a radical shift from TMs. In other words, to what extent is "interaction more powerful than algorithm" (Wegner, 1997)? In fact, Wegner's claim is sharp. In contrast with Milner, Wegner's focus is not on concurrency between computing processes. Instead, he focuses on the complexity of the triggering of external events outside the machine: "Interactive systems are grounded in an external reality both more demanding and richer in behaviour than the rule-based world of non-interactive algorithm" (Wegner, 1997). He strikes the difference between closed and opened systems, the latter being impossibly wholly described. This impossibility makes interactive systems mathematically problematic: they lack completeness.

> The comfortable completeness and predictability of algorithms is inherently inadequate in modelling interactive computing tasks and physical systems. The sacrifice of completeness is frightening to theorists who work with formal models like Turing machines [. . .]. But incomplete behaviour is comfortably familiar to physicists and empirical model builders. Incompleteness is the essential ingredient distinguishing interactive from algorithmic models of computing and empirical from rationalist models of the physical world (Wegner, 1997).

From this, Wegner concludes that computing systems should not be thought of as algorithms but as *interfaces*, *views*, and *modes of use*, definable as behaviours to be specified. Consequently, an ontological question is also at stake: in what terms should the external world be modelled: as atomic objects and events? As processes and flow? Formally, Wegner's account of interaction has led to the development of Persistent Turing machines (PTMs), a model of sequential computation, and the result that multi-stream interaction machines (MIMs) are more expressive than sequential interaction machines (SIMs) (Goldin, 2000; Goldin, Smolka et al., 2004). Wegner and Goldin trace back the idea that interaction is not expressible by or reducible to algorithms at the closing conference on the 5th-generation computing project in the context of logic programming. Reactiveness of logic programs, realized by the commitment to a course of action, was shown to be incompatible with logical completeness (Wegner and Goldin, 1999).

3.3 Legacy

Wegner's work has been criticized, the main objection being that interaction machines can be proved equivalent to TMs. The objections are focused on the defence of the Church-Turing thesis (Cockshott and Michaelson, 2007; Prasse and Rittgen, 1998), and assume that

introducing an interactive computing paradigm denies the results of Church and Turing's work. But this assumption cannot be taken for granted: no one denies that TMs and lambda calculus account for *effective computation*. Both formalisms define the intuitive notion of an *algorithm*. The Church-Turing thesis will only be shaken once someone presents an alternative formal account of an effective procedure. Due to semantic ambiguities, some have interpreted Wegner's work as challenging the Church-Turing thesis. First, Wegner characterizes interaction as *more powerful* than algorithms and TMs. What "powerfulness" precisely refers to is unclear. We will say more about this in the next section (section 4).

Second, there seems to be another semantic ambiguity or alleged identity between "computing" and "computation": "Wegner (and Eberbach) say that it is impossible to describe all computations by algorithms. Thus, they do not accept the classic equation of algorithm and effective computation" (Cockshott and Michaelson, 2007). In the former quoted sentence, a core assumption uses interchangeably "computation" and "computing". But Wegner means that it is impossible to describe everything in computing by algorithms. By "computing", he is referring to what computers do broadly, not to Turing computation in a narrow sense. Therefore, the conclusion made in the quoted sentence does not follow: the identity between an effective computation and an algorithm is not put into question by Wegner.

3.4 Issues for an account of current computers

We are interested in the way Wegner broadens the notion of *interaction*. It is not strictly referring to communicating processes within a computing machine. Possible complex interactions with the environment and the dynamic between inputs and outputs during execution

are considered. However, although debunking the focus of the CTT by stating that interaction is more *powerful* and *expressive* than algorithms, Wegner's work is enclosed in a field of discussion framed by the theory of computability. Furthermore, we still need a way of describing the very mechanisms we are interested in to be provided with a mechanistic account of current computers. This is no surprise since Wegner's work aims primarily to reflect on the theoretical limits of classic mathematical tools, e.g., on notions like *completeness*.

4. Why the interactive models identified do not provide us with an answer

We have reviewed the conceptualization of interactive systems in theoretical computer science. We want to defend that these approaches cannot answer the epistemic question asked by philosophers about what current computers are. There are two reasons for this. First, as we have seen, these conceptualizations focus on whether a formal model for interaction is irreducible to a Turing machine and, if so, whether this is a threat to the Church-Turing thesis. This deprives us of a level of description to explain the mechanisms that allow a computing system to be interactive. We propose to detail here in section 4 the problems posed by the debate on reducibility. We end the section by mentioning a distinction currently offered in the literature that highlights the limits of a formal approach. It is a distinction, mostly worked by Miłkowski, opposing *mechanistic computational explanation* and *model of computation*.

	Interaction as concurrent communicating systems	Interaction as extended Turing Machines	Interaction as a new paradigm
Motivation	Provides new logical foundations to fit new engineering challenges, especially concurrency	Extends the TM model to account for interactive devices	Debunks the strong Church-Turing thesis Discusses the scope of algorithmic solving Prones the need for a new computing paradigm
Account for interaction	Information flow Heterarchical design No complete prediction about overall behavior No end-result Process calculi	External data needed during computation Non-uniformity of programs Interaction with agents Infinity of operations Interactive machines are TMs with advice	Computers have rich interaction with the environment during computing execution, but this processing is not merely algorithmic
Uses and criticisms	First conceptualization of interaction Legacy for automata theory	Inspires the need for a new paradigm Puts at the forefront the Church-Turing thesis	Controversy about the powerfulness of the TM
Issues for an account of interactive computing	Definition of interaction restricted to specific properties: concurrency and communication	Formal oracles cannot account for the physical possibility of entering new data	Issues about powerfulness and expressiveness constrict the debate in the realm of computability theory

Table 1: Sum-up: an overview of explicit theories of interactive computing systems in theoretical computer science.

4.1 Unclear stance towards interaction and Turing reducibility

The first problem with the focus on Turing reducibility in the accounts for interaction is that the stance is not always clear-cut. Milner's work leaves us with the following question: to what extent are the new "logical foundations" for interaction distinct from the classic framework? Irreducibility is not stated in the speech for the Turing Award. There

is a simple translation of lambda-calculus into pi-calculus, which is faithful to computational behaviour. Thus, pi-calculus supports functional programming at a higher level of explanation. However, it is unclear whether any behaviour expressed in the pi-calculus can be translated into a classic calculus. In a more recent book, *The Space and Motion of communicating agents* (2009), Milner introduces bigraphs as another formalism for interactive systems. Bigraphs are proven to have the same expressiveness as Turing machines. It looks like Milner proposes to revise the principle of Occam's razor and praise the plurality of formalisms, models, and frames of explanation:

> I reject the idea that there can be a unique conceptual model, or one preferred formalism, for all aspects of something as large as concurrent computation, which is in a sense the whole of our subject — containing sequential computing as a well-behaved special area. We need many levels of explanation: many different languages, calculi, and theories for the different specialisms (Milner, 1993).

It looks like interaction is the new "basic notion":

> Now, what are the new particles, parts of speech, or elements which allow one to express interaction? They lie at the same elementary level as the operation of a Turing machine on its tape, but they differ. For much longer than the reign of modern computers, the basic idiom of algorithm has been the asymmetric, hierarchical notion of operator acting on operand. But this does not suffice to express interaction between agents as peers; worse, it locks the mind away from the proper mode of thought (Milner, 2006).

As for the work on extended Turing Machines, does it involve that interaction is something else, something irreducible to TMs? Does interaction amount to a classical model of computation with extended

computational power? The latter claim is possibly controversial by revising the Church-Turing thesis. In the end, it looks like interaction is still understood in reference to the classical framework (our italics): "examples of interactive [...] indicate that the classical Turing machine paradigm should be *revised (extended)* in order to capture the forms of computation that one observes in the systems and networks in modern information technology" (Van Leeuwen and Wiedermann, 2000).

Criticisms against Wegner show that the criterion of powerfulness is ambiguous when evaluating a model for a computing system. Does powerfulness refer to computational power, involving that an interactive model can express uncomputable functions in Turing's sense? Or does it refer to the expression of more phenomena? Such ambiguity could support some misunderstanding about interaction.

In any case, the literature review on explicit theories of interaction shows that arguments about the powerfulness and equivalence of the interactive and classic models systematically arise.

4.2 Powerfulness and expressiveness: possible ambiguities

Ambiguities around the concepts of powerfulness and expressiveness likely make the debate need clarification. Indeed, there are at least two ways of understanding them. In any case, the powerfulness of a model refers to its expressiveness, which is a semantic property. Expressiveness refers to *what can be expressed* by a given model. If one thinks of a model as a formal language, let us say that expressiveness relates to all the possible sentences one can make in that language.

In a first sense, powerfulness and expressiveness can be understood strictly within computability theory. In that case, the two notions are used when evaluating a mathematical framework supporting the

formalization of semantics. What is called "powerfulness" refers to *computational power*, and expressiveness refers to a formal criterion evaluating which functions can be expressed. *Turing completeness* is then a possible evaluation criterion for expressiveness, for instance.

Let us say that among the things that could be expressed in a model are functions (*set A*) and other things than functions (*set B*). Within each set, some sets include more than others. Within *set A*, the set of hypercomputations is more expressive than the set of computable functions since it includes the uncomputable ones. That is a way to be more expressive: expressing more functions. However, framing expressiveness and powerfulness as possibly only about computable functions would seem odd to engineers and computer scientists familiar with other formalisms than those related to computability theory. Nevertheless, objections about interaction theories frame the debate in reference to computability theory.

In a second sense, one can consider the powerfulness and expressiveness of a model *outside the strictly formal computability framework*. Since a model must represent, according to specific objectives, a phenomenon of reality or, say, a system, we can understand the powerfulness of a model as a good match between the model and what is modeled.

Therefore, in that broader sense, a model is expressive, given some purpose, if and only if it describes all phenomena required for that given purpose. In that case, the value of the model and concerns about its expressiveness depend on stated goals. From an engineering perspective, for example, a model is valuable to the extent that it allows engineers to think of future systems design easily. In this case, the value of the model could be evaluated, e.g., in terms of usability (effectiveness, efficiency, and satisfaction (ISO, 2018)). From a scientific perspective, the aim is to make good predictions about a system. The

two perspectives are rarely used in isolation since good engineering design requires some science, and good science often relies today on some engineering (Lee, 2017). From the perspective of the philosophy of science and given scientific explanation standards, a good model for a phenomenon rightly describes the mechanisms at stake (Glennan, 2002; Machamer, Darden and Craver, 2000; Miłkowski, 2016). Of course, other possible values for models, from other perspectives, could be found.

To go back to Wegner (Goldin, 2000; Wegner, 1995; 1997; 1998), we argue that this distinction between a narrow and broad sense of expressiveness clarifies criticisms made against him.

In a broad sense, one can interpret Wegner's new paradigm as follows: Wegner considers his interactive model more expressive than a TM by having his model describe *other things than Turing computations*. Wegner's model could then describe more phenomena than a TM. It would not go against the Church-Turing thesis, which remains valid to account for algorithmic problem-solving through effective procedures.

But in a narrow sense of expressiveness, one can interpret (wrongly, we think) the possibility of a new paradigm as follows. Wegner and the tenants of Reactive Turing Machines could think of their interactive model as more expressive than a TM, allowing their model to execute *more functions*, even some of them being uncomputable functions in the sense of the Church-Turing thesis, solving the halting problem. In that case, the claim would indeed be controversial. The bold claim would be the following: a TM is not only providing an account for algorithmic problem-solving through effective procedures but it could also be extended to account for other non-algorithmic processes, solving the uncomputable. Interaction would be some super-calculus, extending the calculative power of

the original TM to account for interaction. It would be satisfactorily modeled with a TM, only given more calculation power. It would go down the track of Accelerating Machines or Super-Turing Machines, able to calculate more than Turing's computable functions (Copeland, 2002; Copeland and Shagrir, 2011; MacLennan, 2009).

We argue that a theory of interaction does not need to embrace the hypercomputation view. Part of an interaction model could be reduced to the classical TM, but some extra elements needed to express interaction cannot be reduced to an a-machine. That does not mean interactive models have super computational power to solve undecidable problems. It simply means interactive systems do things that a TM cannot do. It is possible to admit they do other things *without implying they compute uncomputable functions*.

4.3 What formal models of computation cannot do: providing a mechanistic explanation of computing

So, do we have a good theory about interactive computers? Do we understand what they are? A natural and common way to go is to reduce the question of what interactive computers are to what interactive computation is. Initially, the first models of computation emerged through computability theory. They served as answers to an abstract mathematical problem, namely the formalization of the intuitive notion of an algorithm. They had nothing to say about computers, as computers did not even exist at the time. Since the computability era, models of computation like the Turing Machines have been exported outside their original scope to serve as a basis for theoretical computer science. Some models of computation (Turing Machines) have even helped to reflect on computers. It is no surprise since computers were thought to be precisely the kind of machines that implement computations. Models of computation have then evolved, account-

ing for new desired properties to be integrated within the classical framework. In computer science, what makes a model of computation valuable is related to the formal properties it expresses. Once those formal properties are at hand, they allow further procedures to be acted upon them, especially system verification and certification. In the end, models of computation serve as tools used to support and verify a system's design. These models belong to a particular abstraction level: they do not intend to model the system as a whole and the way it works. They focus on verifiable properties, upon which proofs that guarantee the outputs of the system are built. Verifying formal properties is different from investigating why the system behaves the way it does. They are two different tasks. The former task (verification) belongs to applied mathematics. It describes abstract computations through formal models by focusing on specific properties. The latter (understanding computing behaviour) is the question the philosopher begs when asking what a computer is. It requires something else than task-oriented formalizations of properties abstracted away from any physical mechanism. Philosophers of computing need to make sense of the overall behaviour, which requires combining other levels of abstraction. The reason is that an account of computing behaviour calls upon the description of how computation can be carried out: in other words, it requires the description of execution on some computer architecture. Computations and their models belong to a level of abstraction independent from implementation detail. Computations, as already coined, are "medium-independent" (Klein, 2020). On the contrary, to have a model of some execution belongs to a lower level of abstraction, where minimal references to the devices that allow the execution is made. There is no need to dig into fine-grained implementation details to make sense of computing behaviour in mechanistic terms.

The formal debate on model equivalence and powerfulness leaves us needing more building blocks to figure out an explanation for interactive computing: what makes it possible, and what mechanisms support it? A helpful distinction here capturing why we lack the right tools is a recent distinction in the literature between models of computation (formal) and mechanistic explanations of computing. It deserves attention in the context of understanding interactive computing. Questioning model equivalence belongs to formal mathematics; it does not aim at providing a mechanistic account of the computing phenomena. Interactive models of computation propose an upper layer of abstractions to formalize specific properties but do not hint at how interactive computation is carried out. We suggest we need to adopt a different explanatory focus, departing from the perspective adopted by models of computation and understanding *how interactive computation can be executed*.

Such lessons have just started to be drawn. They have motivated, for example, distinctions between computational models and computational explanations (Klein, 2020) or between models of computation and computational mechanisms (Miłkowski, 2014). The lesson drawn is that formal models of computing systems do not provide us with the appropriate and complete level of description to build an explanation, which is expected to identify the relevant mechanisms at stake. More precisely, an explanation for computing phenomena requires bridging a high-level description of a computation and its blueprint (Miłkowski, 2011; 2016). The approach is based on the standard of mechanistic explanation in science, coupled with the idea that a computational process is intrinsically mechanistic:

> Computational explanations, according to the mechanistic account are constitutive mechanistic explanations: they explain how a mechanism's computational capacity is generated by

> the orchestrated operation of its component parts. To say that a mechanism implements a computation is to claim that the causal organization of the mechanism is such that the input and output information streams are causally linked and that this link, along with the specific structure of information processing, is completely described (Miłkowski, 2014).

If one is looking for a mechanistic explanation of a computing process, Miłkowski argues that a model of computation may be insufficient. The reason something is missing is that a model of computation is not strongly equivalent to a mechanism:

> There are two ways in which computational models may correspond to mechanisms: first, they may be *weakly equivalent* to the explanandum phenomenon, in that they only describe the input and output information, or *strongly equivalent*, when they also correspond to the process that generates the output information (Miłkowski, 2016).

The difference between strong and weak equivalence captures a difference in causal completeness. The formal models of computation are on the side of models that are weakly equivalent to a mechanism: "formal models cannot function as complete causal models of computers. For example, to repair an old broken laptop, it is not enough to know that it was (idealizing somewhat) formally equivalent to a universal Turing machine." (Miłkowski, 2016). An example helps to flesh out the need for such distinction and turns again to the Turing machine:

> Turing machines were not invented to be implemented physically at all, but some people still build them for fun. [...] Imagine a physical instantiation of a trivial logical negation Turing machine, built of, say, steel and rubber and printing

> symbols on paper tape. Its alphabet of symbols consists of "F" and "T." If the machine finds "T" on its tape, it rewrites it to "F" and halts; if it finds "F," it rewrites it to "T" and halts. Let us suppose that the machine's head is so old and worn out that it tears the paper tape during the readout. As a result, no symbol will appear. [. . .] Only when we describe the Turing machine literally, as a causal system that has a particular causal blueprint (engineering specifications of how it is built), can we causally predict such a breakdown. [. . .] Why are breakdowns and malfunctions so important? They help us discover the causal complexity of the system. [. . .] an abstract model of computation will not predict all the possible outcomes of the breakdown, as it abstracts away from a number of the system's causal characteristics. So it will not tell us what is going to happen with the head; it will only say that the computation will no longer be correct (Miłkowski, 2011).

Thus, Miłkowski invites us to consider a new project in the philosophy of computing: "it is necessary to acknowledge the causal structure of physical computers that is not accommodated by the models used in computability theory" (Miłkowski, 2011). To the best of our knowledge, such a project to account for interactive computing has still not been carried out to flesh out the mechanisms at stake. If philosophers of computing were to proceed in that direction, two criteria for a good explanation of a computer proposed by Miłkowski could offer some guidance. First, such an explanation should be complete, in the sense of a complete causal model where causally relevant parts and operations are specified (Miłkowski, 2014). Second, a good explanation for computing should explain the competence of the system: "By providing the instantiation blueprint of the system, we explain the physical exercise of its capacity, or competence, abstractly specified in the formal model" (Miłkowski, 2014). For example, it would be necessary to be able to explain in mechanistic terms what the behaviour

of an oracle corresponds to. This would be equivalent to explaining which mechanisms allow data arrival, launching, interrupting, or pausing machine processes.

Conclusion

We started from the need to update a question in the philosophy of computing: *what is a computer*? Today's computers are highly interactive, so the question can be rephrased more precisely: *what is an interactive computing system*? It is common to understand computers in terms of existing models of computation, hypothesizing that a computer is primarily a machine that carries out computation. Therefore, the working hypothesis has traditionally answered the initial question by asking what computation is. As already noted, this shift should not be taken for granted. There are, however, and the length of the paper does not allow it, historical and epistemological reasons for this shift that have been described and discussed (Daylight, 2014; Haigh and Priestley, 2020; Mol, 2018). We have chosen in this paper to ask ourselves if the shift is relevant in the case of interactive computing: do we understand what an interactive computer is by questioning the formal models proposed in theoretical computer science for interaction? Our literature review shows that there are better paths. There are two reasons for this. First, the conceptualization of interactive systems in theoretical computer science has focused on their comparison with the Turing machine (and sometimes other classical models), putting forward formal questions about powerfulness and equivalence of models that do not clarify the singularity of interactive systems from an epistemic point of view (rather than formal). There is an inherent difficulty in looking for an explanation of a computing

phenomenon in a formal model: it needs more bricks to describe the mechanisms at stake, at a level of abstraction operating the junction between a high-level formalism and the blueprint. This work does not lead us to an aporia but to a research program in the philosophy of computing: we must produce the right level of explanation for interactive computing.[4] This implies an identification of the mechanisms at play that make possible the interaction between processes within the computing machine (whether there are to be thought of as physical or computational processes, or a mix of both[5]) and the environment. The components of such a mechanism are to be identified and described at a level of abstraction that allows a satisfactory reference to the implementation.

Bibliography

Andersen, H.R., Mørk, S. and Sørensen, M.U., 1997. A universal reactive machine. In: A. Mazurkiewicz and J. Winkowski, eds. *Concur '97: concurrency theory. concur 1997* [Online]. Vol. 1243, Lecture notes in computer science. Berlin; Heidelberg: Springer, pp.89–103. https://doi.org/10.1007/3-540-63141-0_7.

Arbach, Y., Karcher, D., Peters, K. and Nestmann, U., 2015. Dynamic causality in event structures. In: S. Graf and M. Viswanathan, eds. *Formal techniques for distributed objects, components, and systems* [Online]. Vol. 9039, Lecture notes in computer science, pp.83–97. https://doi.org/10.1007/978-3-319-19195-9_6.

Aspray, W., 1990. *John von neumann and the origins of modern computing*. Cambridge, MA: MIT Press.

[4] More considerations on such a research program are fleshed out in (Martin, Magnaudet and Conversy, forthcoming).

[5] The distinction between computational and physical processes is out of the scope of this paper but more on this can be found e.g., in (Kycia and Niemczynowicz, 2020).

Backus, J., 1978. Can programming be liberated from the von Neumann style? *Communications of the ACM*, 21(8), pp.613–641.

Baeten, J.C., Luttik, B. and Tilburg, P.V., 2012. Turing meets milner. *Concur'12: proceedings of the 23rd international conference on concurrency theory* [Online], Lecture notes in computer science. Berlin; Heidelberg: Springer, pp.1–20. https://doi.org/10.1007/978-3-642-32940-1_1.

Baeten, J.C., Luttik, B. and Tilburg, P.V., 2013. Reactive Turing machines. In: O. Owe, M. Steffen and J. Telle, eds. *Fct 2011: fundamentals of computation theory* [Online], Lecture notes in computer science. Berlin; Heidelberg: Springer, pp.348–359. https://doi.org/10.1016/j.ic.2013.08.010.

Balcázar, J.L., Díaz, J. and Gabarró, J., 1995. *Structural complexity I.* [Online]. Second Edition, *Texts in Theoretical Computer Science. An EATCS Series*. Berlin; Heidelberg: Springer. https://doi.org/10.1007/978-3-642-97062-7.

Baldan, P., Corradini, A. and Montanari, U., 2001. Contextual petri nets, asymmetric event structures, and processes. *Information and Computation* [Online], 171(1), pp.1–49. https://doi.org/10.1006/inco.2001.3060.

Basman, A., Tchernavskij, P., Bates, S. and Beaudouin-Lafon, M., 2018. An anatomy of interaction: co-occurrences and entanglements. *Conference companion of the 2nd international conference on art, science, and engineering of programming* [Online]. Association for Computing Machinery, pp.188–196. https://doi.org/10.1145/3191697.3214328.

Beaudouin-Lafon, M., 2006. Human-computer interaction. In: D. Goldin, S.A. Smolka and P. Wegner, eds. *Interactive computation: the new paradigm* [Online]. Berlin; Heidelberg: Springer, pp.227–254. https://doi.org/10.1007/3-540-34874-3_10.

Bielecki, A., 2019. *Models of neurons and perceptrons: selected problems and challenges* [Online]. Vol. 770, *Studies in Computational Intelligence*. Springer International Publishing. https://doi.org/10.1007/978-3-319-90140-4.

Cockshott, P. and Michaelson, G., 2007. Are there new models of computation? Reply to Wegner and Eberbach. *Computer Journal* [Online], 50(2), pp.232–247. https://doi.org/10.1093/comjnl/bxl062.

Copeland, B.J., 2002. Hypercomputation. *Minds and Machines* [Online], 12, pp.461–502. https://doi.org/10.1023/A:1021105915386.

Copeland, B.J. and Shagrir, O., 2011. Do accelerating Turing machines compute the uncomputable? *Minds and Machines* [Online], 21, pp.221–239. https://doi.org/10.1007/s11023-011-9238-y.

Daylight, E.G., 2014. A Turing tale. *Communications of the ACM* [Online], 57(10), pp.36–38. https://doi.org/10.1145/2629499.

Dearden, A.M. and Harrison, M.D., 1997. Abstract models for HCI. *International Journal of Human Computer Studies* [Online], 46(1), pp.151–177. https://doi.org/10.1006/ijhc.1996.0087.

Dodig-Crnkovic, G., 2011. Significance of models of computation, from turing model to natural computation. *Minds and Machines* [Online], 21, pp.301–322. https://doi.org/10.1007/s11023-011-9235-1.

Eberbach, E., Goldin, D. and Wegner, P., 2004. Turing's ideas and models of computation. In: C. Teuscher, ed. *Alan turing: life and legacy of a great thinker* [Online]. Berlin; Heidelberg: Springer, pp.159–194. https://doi.org/10.1007/978-3-662-05642-4_7.

Glabbeek, R.V. and Plotkin, G., 2004. Event structures for resolvable conflict. *29th international symposium on mathematical foundations of computer* [Online]. Vol. 3153, Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics). Berlin; Heidelberg: Springer, pp.550–561. https://doi.org/10.1007/978-3-540-28629-5_42.

Glennan, S., 2002. Rethinking mechanistic explanation. *Philosophy of Science* [Online], 69(S3), pp.342–353. https://doi.org/10.1086/341857.

Godfrey, M.D. and Hendry, D.F., 1993. The computer as von neumann planned it. *IEEE Annals of the History of Computing* [Online], 15(1), pp.11–21. https://doi.org/10.1109/85.194088.

Goldin, D., 2000. Persistent Turing machines as a model of interactive computation. In: K. Schewe and B. Thalheim, eds. *Foundations of Information and Knowledge Systems. FoIKS 2000* [Online]. Vol. 1762, Lecture notes in computer science. Berlin; Heidelberg: Springer, pp.116–135. https://doi.org/10.1007/3-540-46564-2_8.

Goldin, D., Smolka, S.A., Attie, P.C. and Sonderegger, E.L., 2004. Turing machines, transition systems, and interaction. *Information and Computation* [Online], 194(2), pp.101–128. https://doi.org/https://doi.org/10.1016/j.ic.2004.07.002.

Goldin, D. and Wegner, P., 2008. The interactive nature of computing: Refuting the strong Church-Turing thesis. *Minds and Machines* [Online], 18, pp.17–38. https://doi.org/10.1007/s11023-007-9083-1.

Goldin, D., Wegner, P. and Smolka, S.A., 2006. *Interactive computation: the new paradigm* [Online]. Berlin; Heidelberg: Springer. https://doi.org/10.1007/3-540-34874-3.

Haigh, T. and Priestley, M., 2020. Historical reflections von neumann thought turing's universal machine was 'simple and neat.' but that didn't tell him how to design a computer. *Communications of the ACM* [Online], 63(1), pp.26–32. https://doi.org/10.1145/3372920.

Harel, D. and Pnueli, A., 1985. On the development of reactive systems. In: K. Apt, ed. *Logics and models of concurrent systems* [Online]. Vol. 13, Nato asi series. Berlin; Heidelberg: Springer, pp.477–498. https://doi.org/10.1007/978-3-642-82453-1_17.

Hartmanis, J., 1994. Turing award lecture on computational complexity and the nature of computer science. *Communications of the ACM* [Online], 37(10), pp.37–43. https://doi.org/10.1145/194313.214781.

Hornbaek, K. and Oulasvirta, A., 2017. What is interaction? *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* [Online]. New York, NY: Association for Computing Machinery, pp.5040–5052. https://doi.org/10.1145/3025453.3025765.

ISO, 2018. Ergonomics of human-system interaction. part 11: usability: definitions and concepts. *ISO 9241-11:2018*.

Klein, C., 2020. Polychrony and the process view of computation. *Proceedings of the 2018 Biennial Meeting of the Philosophy of Science Association. Part II* [Online]. Vol. 87, 5, pp.1140–1149. https://doi.org/10.1086/710613.

Knuth, D.E., 1968. *The art of computer programming, vol.1: fundamental algorithms*. Boston; etc.: Addison-Wesley.

Kycia, R. and Niemczynowicz, A., 2020. Information and physics. *Philosophical Problems in Science (Zagadnienia Filozoficzne w Nauce)* [Online], (69), pp.237–252. Available at: <https://zfn.edu.pl/index.php/zfn/article/view/513>.

Lee, E.A., 2017. Fundamental limits of cyber-physical systems modeling. *ACM Transactions on Cyber-Physical Systems* [Online], 1(1), pp.1–26. https://doi.org/10.1145/2912149.

Lee, E.A., 2020. *Plato and the nerd* [Online]. MIT Press. https://doi.org/10.7551/mitpress/11180.001.0001.

Lee, E.A. and Neuendorffer, S., 2006. Concurrent models of computation for embedded software. *System-on-Chip: Next Generation Electronics* [Online]. https://doi.org/10.1049/PBCS018E_ch7.

Lee, E.A. and Sangiovanni-Vincentelli, A., 1998. A framework for comparing models of computation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* [Online]. https://doi.org/10.1109/43.736561.

Luttik, B. and Yang, F., 2016. On the executability of interactive computation. In: A. Beckmann, L. Bienvenu and N. Jonoska, eds. *Pursuit of the universal. cie 2016* [Online]. Vol. 9709, Lecture notes in computer science. Cham: Springer, pp.312–322. https://doi.org/10.1007/978-3-319-40189-8_32.

Machamer, P., Darden, L. and Craver, C.F., 2000. Thinking about mechanisms. *Philosophy of Science* [Online], 67(1), pp.1–25. https://doi.org/10.1086/392759.

MacLennan, B., 2003. Transcending Turing computability. *Minds and Machines* [Online], 13, pp.3–22. https://doi.org/10.1023/A:1021397712328.

MacLennan, B., 2009. Super-Turing or non-Turing? extending the concept of computation. *International Journal of Unconventional Computing*, 5(3-4), pp.369–387.

Manna, Z. and Pnueli, A., 1992. *The temporal logic of reactive and concurrent systems* [Online]. Springer. https://doi.org/10.1007/978-1-4612-0931-7.

Martin, A., Magnaudet, M. and Conversy, S., forthcoming. Computers as interactive machines: Can we build an explanatory abstraction? *Minds and Machines*.

Milner, R., 1975. Processes: A Mathematical Model of Computing Agents. In: H.E. Rose and J.C. Shepherdson, eds. *Studies in Logic and the Foundations of Mathematics* [Online]. Vol. 80, *Logic Colloquium '73*. Elsevier, pp.157–173. https://doi.org/10.1016/S0049-237X(08)71948-7.

Milner, R., 1982. Four combinators for concurrency. *Proceedings of the annual acm symposium on principles of distributed computing* [Online], Podc '82. New York, NY: Association for Computing Machinery, pp.104–110. https://doi.org/10.1145/800220.806687.

Milner, R., 1983. Calculi for synchrony and asynchrony. *Theoretical Computer Science* [Online]. https://doi.org/10.1016/0304-3975(83)90114-7.

Milner, R., 1993. Elements of interaction: Turing Award Lecture. *Communications of the ACM* [Online], 36(1), pp.78–89. https://doi.org/10.1145/151233.151240.

Milner, R., 1999. *Communicating and mobile systems: the π-calculus*. Cambridge: Cambridge University Press.

Milner, R., 2006. Turing, computing and communication [Online]. In: ed. by D. Goldin, S.A. Smolka and P. Wegner. Berlin; Heidelberg: Springer, pp.1–8. https://doi.org/10.1007/3-540-34874-3_1.

Milner, R., 2009. *The space and motion of communicating agents*. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511626661.

Miłkowski, M., 2011. Beyond formal structure: a mechanistic perspective on computation and implementation. *Journal of Cognitive Science*, 12, pp.359–379.

Miłkowski, M., 2014. Computational mechanisms and models of computation. *Philosophia Scientiae* [Online], 18(3), pp.215–228. https://doi.org/10.4000/philosophiascientiae.1019.

Miłkowski, M., 2016. A mechanistic account of computational explanation in cognitive science and computational neuroscience. In: V.C. Müller, ed. *Computing and Philosophy: Selected Papers from IACAP 2014* [Online]. Vol. 375. Springer International Publishing, pp.191–205. https://doi.org/10.1007/978-3-319-23291-1_13.

Mol, L.D., 2018. Turing machines. *Stanford Encyclopedia* [Online]. Available at: <https://plato.stanford.edu/entries/turing-machine/>.

Myers, B., 1994. Challenges of HCI design and implementation. *Interactions* [Online]. https://doi.org/10.1145/174800.174808.

Nielsen, M., Plotkin, G. and Winskel, G., 1981. Petri nets, event structures and domains, part I. *Theoretical Computer Science* [Online], 13(1), pp.85–108. https://doi.org/10.1016/0304-3975(81)90112-2.

Nisan, N. and Schocken, S., 2005. *The Elements of Computing Systems: Building a Modern Computer from First Principles*. Cambridge: MIT Press.

Petri, C., 1980. Introduction to general net theory. In: W. Brauer, ed. *Net theory and applications* [Online]. Vol. 84, Lecture notes in computer science. Berlin; Heidelberg: Springer. https://doi.org/10.1007/3-540-10001-6_21.

Piccinini, G., 2008. Computers. *Pacific Philosophical Quarterly* [Online], 89(1), pp.32–73. https://doi.org/10.1111/j.1468-0114.2008.00309.x.

Pierce, B.C. and Turner, D.N., 2000. Pict: a programming language based on the pi-calculus. *Proof, Language and Interaction: Essays in Honour of Robin Milner*. Cambridge, MA: MIT Press, pp.455–494.

Pour-El, M.B., 1999. The structure of computability in analysis and physical theory: an extension of Church's thesis. In: E. Griffor, ed. *Handbook of Computability Theory, Studies in Logic and the Foundations of Mathematics* [Online]. Amsterdam: Elsevier, pp.449–471. https://doi.org/10.1016/S0049-237X(99)80029-9.

Prasse, M. and Rittgen, P., 1998. Why Church's thesis still holds. some notes on Peter Wegner's tracts on interaction and computability. *The Computer Journal* [Online], 41, pp.357–362. https://doi.org/10.1093/comjnl/41.6.357.

Rapaport, W.J., 2018. What is a computer? a survey. *Minds and Machines* [Online], 28(3), pp.385–426. https://doi.org/10.1007/s11023-018-9465-6.

Rogers, H., 1987. *Theory of recursive functions and effective computability*. Cambridge, MA: MIT Press. https://doi.org/10.2307/3614588.

Siegelmann, H.T., 1995. Computation beyond the Turing limit. *Science* [Online], 268(5210), pp.545–548. https://doi.org/10.1126/science.268.5210.545.

Smith, B.C., 2002. The foundations of computing. In: M. Scheutz, ed. *Computationlism: new directions*. Cambridge, MA: MIT Press.

Soare, R.I., 2013. Interactive computing and relativized computability. In: *Computability: Turing, Godel, Church, and Beyond* [Online]. Ed. by B.J. Copeland. Cambridge, MA: MIT Press. Chap. 9, pp.214–271. https://doi.org/10.7551/mitpress/8009.003.0010.

Staiger, L., 1997. Omega-languages. In: *Handbook of formal languages* [Online]. Ed. by G. Rozenberg and A. Salomaa. Berlin; Heidelberg: Springer, pp.339–387. https://doi.org/10.1007/978-3-642-59126-6_6.

Thomas, W., 1990. Automata on infinite objects. In: *Formal models and semantics* [Online]. Amsterdam: Elsevier. Chap. 4, pp.133–191. https://doi.org/10.1016/b978-0-444-88074-1.50009-3.

Turing, A., 1937. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society* [Online], s2-42(1), pp.230–265. https://doi.org/10.1112/plms/s2-42.1.230.

Van Leeuwen, J. and Wiedermann, J., 2000. On the power of interactive computing. In: J. van Leeuwen et al., eds. *Theoretical computer science: exploring new frontiers of theoretical informatics* [Online]. Vol. 1872, *TCS 2000. Lecture Notes in Computer Science*. Berlin; Heidelberg: Springer. https://doi.org/10.1007/3-540-44929-9_48.

Van Leeuwen, J. and Wiedermann, J., 2001. Beyond the turing limit: Evolving interactive systems. In: L. Pacholski and P. Ružička, eds. *SOFSEM 2001: Theory and Practice of Informatics* [Online]. Vol. 2234, Lecture notes in computer science. Berlin; Heidelberg: Springer, pp.90–109. https://doi.org/10.1007/3-540-45627-9_8.

Van Leeuwen, J. and Wiedermann, J., 2006. A theory of interactive computation. *Interactive computation: the new paradigm* [Online]. Berlin; Heidelberg: Springer, pp.119–142. https://doi.org/10.1007/3-540-34874-3_6.

Wegner, P., 1995. Interaction as a basis for empirical computer science. *ACM Computing Surveys (CSUR)* [Online], 27(1), pp.45–48. https://doi.org/10.1145/214037.214092.

Wegner, P., 1997. Why interaction is more powerful than algorithms. *Communications of the ACM* [Online], 40(5), pp.80–91. https://doi.org/10.1145/253769.253801.

Wegner, P., 1998. Interactive foundations of computing. *Theoretical Computer Science* [Online], 192(2), pp.315–351. https://doi.org/10.1016/S0304-3975(97)00154-0.

Wegner, P. and Goldin, D., 1999. Interaction as a framework for modeling. In: G. Goos et al., eds. *Conceptual modeling: current issues and future directions* [Online]. Vol. 1565, *Lecture Notes in Computer Science*. Berlin; Heidelberg: Springer, pp.243–257. https://doi.org/10.1007/3-540-48854-5_19.

Wegner, P. and Goldin, D., 2003. Computation beyond Turing machines. *Communications of the ACM* [Online], 46(4), pp.100–102. https://doi.org/10.1145/641205.641235.

Wegner, P. and Goldin, D., 2006. Principles of problem solving. *Communications of the ACM* [Online], 49(7), pp.27–29. https://doi.org/10.1145/1139922.1139942.