Human and Machine Computation: An Exploration

Máté Szabó

Submitted in partial fulfillment of the requirement s for the degree of
Doctor of Philosophy in Logic, Computation and Methodology in
Department of Philosophy at Carnegie Mellon University,
2017

Dissertation Committee:

Wilfried Sieg, Carnegie Mellon University
Jeremy Avigad, Carnegie Mellon University
Stephen Brookes, Carnegie Mellon University
David Danks, Carnegie Mellon University
Liesbeth De Mol, Université de Lille 3

Contents

Introduction

The aim of this Dissertation is to explore human and machine computations and provide their unifying delimitation by the concept of a Computable Dynamical System. That concept is used to examine contemporary models of parallel computation in computer science and of human cognition in psychology. The delimitative unification of these different kinds of computations is achieved through two different kinds of explorations.

The first exploration is a conceptual one. It leads to Sieg's structural axiomatization of the notion of computability. The axiomatic treatment provides a sharp delimitation of the concept while unifying human and machine computation building on Turing's and Gandy's analysis of human and machine computation respectively. This axiomatization is a result of an analysis of the notion of computability itself and provides a structural characterization of it, under which computational devices fall. At the same time, Sieg's representation theorem shows that any Computable Dynamical System is computationally reducible to a Turing Machine.

The second exploration is an applied and deeply interdisciplinary one. It investigates realistic models of machine computation and current models of human cognition as Computable Dynamical Systems. My aim is to show that the great majority of these models are Computable Dynamical Systems. This amounts to a partial unification in the sense that it shows that machine computation and most current models of cognitive processes do fall under the same structural axiomatization of computability. Of course, based on our current understanding of human cognition, a complete unification cannot be claimed. To explore the "borders" of this delimitative unification I take a look at models that might not be Computable Dynamical Systems.

Let us now turn to a more detailed description of the Dissertation. The first, conceptual exploration provides an overview of the analysis of the notion of computability. It starts from the mathematical context of the late 19th century, arriving at Sieg's structural axiomatization of the notion, through the mathematical and logical developments of the 1930s. That is, after Dedekind's work is alluded to, the quest for the characterization of effectively calculable number theoretic functions in the 1930s is examined. Here I survey the work of Gödel, Church, and Hilbert and Bernays, following the development of the notion of general recursiveness. This serves as an intellectual and historical context for the discussion of Turing's (1936). There Turing gave a convincing and satisfactory analysis of the notion of human mechanical computations. Indeed, when Church reviewed Turing's work in 1937, he said that compared to general recursiveness and λ-calculability, Turing machines have "the advantage of making the identification with effectiveness in the ordinary sense evident immediately" (p. 43). Gödel also became convinced by Turing's (1936), but only much later. The novelty and persuasiveness of Turing's analysis resulted from his different perspective: he analyzed the calculations of a human computer directly, taking into account their cognitive limitations. This perspective is indeed quite different from that of his predecessors, who investigated the procedure of

evaluating a function within a calculus.

Though many scholars read Turing's analysis as an analysis of machine computability, that is a clear misinterpretation. That task was taken up only by Gandy, Turing's student and friend. His analysis of machine computability parallels Turing's in the following sense. Where Turing took the cognitive limitations of the human computer into account, Gandy used physical principles to impose limitations on the computing procedures. Part of Gandy's aim was to include parallel computations in his model as well. He showed that even the paradigmatic theoretical computing device, Conway's Game of Life, does fit his conditions. The final step of the conceptual exploration of computability is Sieg's structural axiomatization of the notion, leading to the structural description of Computable Dynamical Systems. Sieg's axiomatization not only builds on Turing's and Gandy's analysis and encompasses their "devices", but generalizes the domain to which computational processes can be applied. While in Turing's case only finite strings are computed with, in Sieg's axiomatization configurations that can be encoded in terms of hereditarily finite sets can be computed with. However, it is important to emphasize that Computable Dynamical Systems are not yet another particular computational device, but an axiomatically characterized notion under which other devices fall. At the same time, all Computable Dynamical Systems are Turing computable.

Many scholars were wishing for a formal axiomatic treatment of computability with the hope that once such an axiomatization is available, Church's Thesis might become provable. In a short section I provide a critical survey of that literature, arguing that Church's Thesis is not a statement that can be decided by a mathematical proof.

This conceptual exploration of previous work serves as the background of my investigations, that is, an exploration of applications. In Section 4, *Models of Parallel Computing as Computable Dynamical Systems*, I show that models of realistic parallel computational devices satisfy Sieg's axioms. Although Gandy's analysis covered machine computations and Sieg's axiomatization encompasses it, realistic parallel computers were not yet shown to satisfy the axioms of Computable Dynamical Systems. As there is no generally accepted single theoretical model of parallel machine computation, I provide a survey of such models proposed in the literature. Then I exemplify how such models can be seen to be Computable Dynamical Systems. This is achieved through a detailed description of an imaginary parallel device that exhibits the features of the parallel computational models surveyed. Finally, the "borders" of machine computability are explored by looking at analogue and quantum computers and touching upon hypercomputing.

The last part, *Models of Cognition as Computable Dynamical Systems*, investigates contemporary cognitive models. The main aim here is to demonstrate that the majority of these models satisfy the axioms of Computable Dynamical Systems as well. Previously De Pisapia showed that artificial neural networks are Computable Dynamical Systems. Since artificial neural networks and models using parallel computations account for a fraction of cognitive models, the task to cover the remaining ones still would be almost inexhaustible if they

would have to be treated one by one. However, David Danks recently proposed a new cognitive architecture in his *Unifying the Mind* and claimed that many current cognitive models can be unified, if it is assumed that the different processes use shared representations in the form of graphical models. I show that graphical models are Computable Dynamical Systems, and work out in detail how to represent heuristic processes of human causal structure learning. At the end the "borders" are explored once again. It is indicated how more involved processes, such as analogical reasoning, could be accommodated in this framework. Finally, some theories are mentioned which may be outside the scope of Computable Dynamical Systems.

Thus, a rather overarching unification of realistic parallel computing devices and current models of cognitive science is provided in terms of Computable Dynamical Systems. At the same time the axiomatic characterization of this notion is used to explore the "borders" of computability.

Acknowledgements

First of all, I would like to thank my advisor Wilfried Sieg. I thank him for his guidance and continuous support of my studies and research in the last six years, and his patience during the stressful stretches of this process. More concretely, I thank him for suggesting the topic of the dissertation, unifying realistic models of parallel computing with models of cognition based on David Danks' work. Furthermore, his work provides an overarching, axiomatic perspective of computability, which made my investigations possible.

I would also like to thank my committee members, Jeremy Avigad, Stephen Brookes, David Danks and Liesbeth De Mol for providing careful feedback and suggestions for my work. Their input made my Dissertation significantly better.

I am indebted to the graduate student body at the Department of Philosophy, for providing a friendly and welcoming atmosphere, and helping me to adjust to another country and culture. I always appreciated the thriving intellectual life and friendships on and off campus. In particular, I would like to thank my friend Patrick Walsh for keeping me grounded.

Finally, I must express my very profound gratitude to my family and Réka Bence, who made it possible for me to start my studies at Carnegie Mellon University.

1 Effective Calculability

The notion of computability is obviously the central notion when investigating human and machine computability. The next section is devoted entirely to the work of Alan Turing from 1936, introducing his machines and analyzing human mechanical computing, Gandy's work from 1980 on analyzing machine and parallel computing, and Sieg's structural axiomatization of the notion of computability, building on and encompassing the work of Turing and Gandy. This short chapter provides the intellectual and historical context for those investigations. That is, the work from the 1930s on mechanical procedures and effective calculability, focusing on the contributions of Gödel, Church, Kleene, and Hilbert and Bernays. As we will see, all of them attempted to characterize mechanical procedures or (effective) calculability of number theoretic functions in terms of restricted calculations within deductive formalisms. For a detailed and exhaustive account of the historical and intellectual context see Sieg's (2009) that incorporates the analytic work done in his papers (1994) and (1997).

1.1 Primitive Recursion

Before turning to the work on recursive functions in the 1930s, the notion of primitive recursion has to be at least mentioned. This notion originates in the foundational work of Dedekind.

In his *Was sind und was sollen die Zahlen?* (1888), Dedekind provided a structural axiomatization of simply infinite systems, now familiar as the Dedekind-Peano axioms of natural numbers. Dedekind proves, in contemporary terminology, the categoricity of the axioms (Theorem 132). In the proof he relies on a previous result, Theorem 126, called *Theorem of the definition by induction*. There he considers a simply infinite system $(N, \varphi, 1)$, where N stands for natural numbers, φ for the successor function, and 1 for unit, and another system Ω with an arbitrary function[1] θ defined on it and an $\omega \in \Omega$. Then he proves that there is exactly one function $\psi : N \to \Omega$ that satisfies the following, *recursion* equations:

$$\psi(1) = \omega,$$
$$\psi(\varphi(n)) = \theta(\psi(n)).$$

Towards the end of his (1888), concerned with actual mathematical practice, Dedekind also defines addition, multiplication, and exponentiation using this recursive schema.

Sieg emphasizes Dedekind's "very abstract idea": showing the existence of a unique solution for a functional equation, as well as the importance of viewing functions to be given by calculation procedures. (2009, p. 541) Both of these aspects became important for later developments, starting with Skolem's and Hilbert's foundational work in the 1920s. By the middle of that decade examples of calculable but not primitive recursive functions were known in Göttingen.

[1] Dedekind calls θ a mapping, where mappings are different sorts than sets.

This lead to a quest for the characterization of the notion of calculable functions. The remainder of this section is devoted to that enterprise, focusing on the work of Gödel, Church, Kleene, and Hilbert and Bernays in the 1930s.

1.2 Gödel on Mechanical Procedures

Gödel's incompleteness results were published in his (1931), entitled *On Formally Undecidable Propositions of* Principia Mathematica *and Related Systems I*. According to him, "[t]he development of mathematics toward greater precision has led [...] to the formalization of large tracts of it, so that one can prove any theorem using nothing but a few mechanical rules." Here Gödel shows that "the most comprehensive formal systems that have been set up" to date, such as *Principia Mathematica* and Zermelo–Fraenkel set theory, contain "relatively simple problems in the theory of integers that cannot be decided on the basis of the axioms." At the same time he emphasizes that "[t]his situation is not in any way due to the special nature of the systems that have been set up, but holds for a wide class of formal systems" (p. 151). However, around 1930-1931 he was not able to provide the most general formulation of his result. In an unpublished draft for a talk from the late 1930s he described the situation as follows:

> When I first published my paper about undecidable propositions the result could not be pronounced in this generality, because for the notions of mechanical procedure and of formal system no mathematically satisfactory definition had been given at that time.[2] (193?, p. 166)

The emphasis here is on mechanical procedures: "The essential point is to define what a [mechanical] procedure is. Then the notion of a formal system follows easily" (p. 166). The latter can simply be understood as a formal language with a finite set of axioms and inference rules whose applicability is decided by mechanical procedures.

Gödel lectured on his incompleteness results at the Institute of Advanced Study in Princeton from February to May in 1934. The title of the lecture notes, *On Undecidable Propositions of Formal Mathematical Systems*,[3] already shows that steps were taken towards a generalization of his results. In the *Introduction* Gödel gives an informal description of the role of "finite procedure" in an attempt to describe the "constructive" character of formal systems:

> We require that the rules of inference, and the definitions of meaningful formulas and axioms, be constructive; that is, for each rule of inference there shall be a finite procedure for determining whether a

[2] Gödel continues by saying that "This gap has since been filled by Herbrand, Church and Turing." The work of Church will be discussed in the next subsection, while Turing's in the next section. On Gödel's connection to Herbrand's work see Sieg's (2003) and (2005).

[3] The lecture notes were taken by Kleene and Rosser and were circulated as mimeographed notes. For more information on the origination of the notes see Kleene's (1986).

> given formula B is an immediate consequence (by that rule) of given formulas $A_1, ..., A_n$, and there shall be a finite procedure for determining whether a given formula A is a meaningful [i.e. well-formed] formula or an axiom. (1934, p. 346)

This informal notion of finite procedure is later sharpened with the mathematical notion of primitive recursion. Indeed, Gödel asserts that "the relation of immediate consequence shall be [primitive][4] recursive." (p. 361) In §2, where the primitive recursive functions and relations are introduced, Gödel clearly emphasizes the computable feature of these functions:

> [Primitive] Recursive functions have the important property that, for each given set of values of the arguments, the value of the function can be computed by a finite procedure. (p. 348)

This description is immediately followed by this footnote:

> The converse seems to be true if, besides recursions according to the scheme (2) [of primitive recursion], recursions of other forms (e.g. with respect to two variables simultaneously) are admitted. This cannot be proved, since the notion of finite computation is not defined, but it serves as a heuristic principle.[5] (p. 348, footnote 3)

It was known by the end of the 1920s that there are functions which are not primitive recursive, yet calculable by finite procedures, such as Ackermann's and Sudan's functions.[6] The possible expansion of the notion of recursion to capture all finite procedures is discussed by Gödel in §9, entitled *General Recursive Functions*. As a natural attempt to generalize the notion, he starts from the following definition:

> If ϕ denotes an unknown function, and $\psi_1, ..., \psi_k$ are known functions, and if the ψ's and ϕ are substituted in one another in the most general fashions and certain pairs of the resulting expressions are equated, then, if the resulting set of functional equations has one and only one solution for ϕ, ϕ is a recursive function. (p. 368)

[4] Here, following the contemporaneous usage, Gödel calls primitive recursive functions simply 'recursive' while recursive functions are called 'general recursive.'

[5] Although it might seem as an early expression of a version of Church's Thesis (see the next subsection), we know from Gödel himself that it was not meant that way: Sieg quotes (2009, p. 558) from Gödel's letter to Martin Davis of 15 February 1965:

> The conjecture stated there only refers to the equivalence of "finite (computation) procedure" and "recursive procedure". However, I was, at the time of these lectures, not at all convinced that my concept of recursion comprises all possible recursions; and in fact the equivalence between my definition and Kleene's ... is not quite trivial.

[6] On the less familiar example of Sudan's see Claude, Marcus and Tevy's (1979).

Gödel imposes two restrictions on this definition, he mistakenly attributes to Herbrand.[7] The first requires the left-hand side of the equations to take the form:

$$\phi(\psi_{i1}(x_1, ..., x_n), \psi_{i2}(x_1, ..., x_n), ..., \psi_{il}(x_1, ..., x_n)).$$

The second restrictive *regularity condition* ensures that ϕ is computable in a calculus. More precisely it requires that for every l-tuple, $k_1, ..., k_l$, of natural numbers there is exactly one natural number m, such that $\phi(k_1, ..., k_l) = m$ is a "derived equation." Where from a set of given functional equations the derived equations are inductively defined as follows:

> (1a) Any expression obtained by replacing all the variables of one of the given equations by natural numbers shall be a derived equation.
>
> (1b) $\psi_{ij}(k_1, ..., k_n) = m$ shall be a derived equation if $k_1, ..., k_n$ are natural numbers and $\psi_{ij}(k_1, ..., k_n) = m$ is a true equality.
>
> (2a) If $\psi_{ij}(k_1, ..., k_n) = m$ is a derived equation, the equality obtained by substituting m for an occurrence of $\psi_{ij}(k_1, ..., k_n)$ in a derived equation shall be a derived equation.
>
> (2b) If $\phi(k_1, ..., k_l) = m$ is a derived equation, where $k_1, ..., k_l, m$ are natural numbers, the expression obtained by substituting m for an occurrence of $\phi(k_1, ..., k_l)$ on the right-hand side of a derived equation shall be a derived equation.

Sieg emphasizes two features of Gödel's definition. One, "the formulation of the *regularity condition* requiring computable functions to be total, but without insisting on a finitist proof [as Herbrand had done]." Although this condition was dropped later, it was an "absolutely crucial step [...] to disassociate general recursive functions from the epistemologically restricted notion of proof". The other feature is the "precise specification of *mechanical* rules for deriving equations, i.e., for carrying out numerical computations" (2009, p. 556). Gödel himself emphasized this feature in letters to van Heijenoort. On the 14th of August 1964 Gödel wrote: "it was exactly by *specifying* the rules of computation that a mathematically workable and fruitful concept was obtained." (van Heijenoort 2003, p. 316) Similarly, on the 23rd of April 1963 he wrote: "the computation, for *all* computable functions, proceeds by *exactly the same rules*. It is this fact which makes a precise notion of general recursiveness possible." (van Heijenoort 2003, p. 308) However, we know that in 1934 Gödel was "not at all convinced" that his "concept of recursion comprises all possible recursions"; see footnote 5.

This clearly changed by the end of the 1930s, as we can see from Gödel's (193?), entitled *Undecidable Diophantine Propositions*. Once again, the notion of mechanical procedure is the focus in order to formulate the incompleteness

[7] Gödel refers to his private correspondence with Herbrand (2003) as well as Herbrand's (1931) when attributing this definition to him. On the intricate history of their interaction, Gödel's recollection, and Herbrand's contributions see Sieg's (2003) and (2005).

results in their most general form. After primitive recursive functions are introduced a need for a broader class of calculable functions is expressed. According to Gödel, "one way [...] of arriving at such a definition is to generalise this scheme of [primitive] recursion. It can be thought of as a definition by postulates" for the functions occurring in the schema of primitive recursion. "Now all these postulates have the following two characteristics" (p. 167). The first characteristic is an inductive definition of a functional calculus (given in an informal way). While the second characteristic is a simplified version of the second restriction from his (1934), a crystallized description of the mechanical rules allowed while computing functions:

> The second characteristic of recursive postulates is that they allow one to calculate the values of the function defined. Now by analyzing in which manner this calculation proceeds you will find that it makes use only of the following two rules:
>
> R1. to substitute particular integers such as 3 or 4 for the variables $x, y, ...$;
>
> R2. to substitute equals for equals, i.e., if you have arrived in the course of your calculation at an equation $T_1 = T_2$, you are allowed to substitute T_1 for T_2 within any other equation which you may obtain. (193?, pp. 167-168)

The more general schema of recursion is immediately followed by the claim that this generalization amounts to characterizing computable functions, and that it provides "their correct definition":

> And now it turns out that these two characteristics are exactly those that give the correct definition of a computable function. (p. 168)

Shortly after, Gödel alludes to Turing's work:

> That this really is the correct definition of mechanical computability was established beyond any doubt by Turing. (p. 168)

According to Gödel, Turing established the correctness of the definition of mechanical computability by showing that the functions defined above can literally be calculated by a machine. That is, one "can construct a machine with a finite number of parts" that will calculate its values once its "crank" is turned.[8] Although Gödel's description of Turing Machines is rather bizarre,[9] it shows that the image of the mechanical machine did play an important role in his conviction. As Sieg formulates, "the implicit claim is clearly that a procedure is mechanical just in case it is executable by a machine with a finite number of parts." (2009, p. 602)

[8] However, Turing did not prove the equivalence of general recursiveness and Turing machine computability directly. He showed its equivalence with λ-definability, which was, in turn, known to be equivalent with recursiveness.

[9] On Gödel's familiarity with and understanding of Turing's work see Sieg's 6.1 of his (2009) and (2013).

Now we turn to the work of Alonzo Church on effective calculability. While Gödel found Turing's analysis of "mechanical procedure" convincing "beyond any doubt", he found Church's idea to identify effective calculability with λ-calculability in 1934 "thoroughly unsatisfactory". The next subsection examines Church's work and Thesis from 1936, while Turing's analysis is explored at the beginning of the next section.

1.3 Church's Thesis

In 1934 when Gödel lectured at the Institute of Advanced Study, Church was working on the λ-calculus with his students, Kleene and Rosser. In the previous subsection we saw that Gödel tried to characterize the notion of mechanical or finite procedures by generalizing the definition of primitive recursion. Church was thinking about identifying the notion of effective calculability with λ-definability already in 1932-1933. Gödel's reaction was reported in a letter of Church's to Kleene, dated November 29, 1935:

> In regard to Gödel and the notions of recursiveness and effective calculability, the history is the following. In discussion [sic] with him the notion of lambda-definability, it developed that there was no good definition of effective calculability. My proposal that lambda-definability be taken as a definition of it he regarded as thoroughly unsatisfactory. (quoted in Davis 1982, p. 9)

Although Church was convinced about the appropriateness of λ-definability, in his classical (1936) he defined effective calculability in terms of recursiveness.[10] In the *Introduction* the aim of the paper is described as follows:

> There is a class of problems of elementary number theory which can be stated in the form that it is required to find an effectively calculable function f of n positive integers, such that $f(x_1, x_2, ..., x_n) = 2$ is a necessary and sufficient condition for the truth of a certain proposition of elementary number theory involving $x_1, x_2, ..., x_n$ as free variables.[11]
>
> [...]
>
> The purpose of the present paper is to propose a definition of effective calculability which is thought to correspond satisfactorily to the somewhat vague intuitive notion in terms of which problems of this

[10] Church's letter to Kleene as quoted by Davis continues with Church's response to Gödel's critique: "I replied that if he would propose any definition of effective calculability which seemed even partially satisfactory I would undertake to prove that it was included in lambda-definability." Church was also right in believing the appropriateness of λ-definability, as it is equivalent with recursiveness and Turing Machine computability. Nevertheless from 1935 he used the notion of recursiveness instead of λ-definability (Church 1935). For a detailed retrospection see Kleene's (1981).

[11] Church remarks in a footnote that "The selection of the particular positive integer 2 instead of some other is, of course, accidental and non-essential."

> class are often stated and to show, by means of an example, that not every problem of this class is solvable. (pp. 345-346)

The definition of effective calculability is then given in §7:

> We now define the notion, already discussed, of an *effectively calculable* function of positive integers by identifying it with the notion of a recursive function of positive integers (or of a λ-definable function of positive integers). This definition is thought to be justified by the considerations which follow, so far as positive justification can ever be obtained for the selection of a formal definition to correspond to an intuitive notion. (p. 356)

This identification of effective calculability with recursiveness was later dubbed by Kleene as *Church's Thesis*. The remainder of §7 of Church's (1936) is devoted to arguing for the Thesis. As it can be inferred from his description, the mathematical argument provided there is not intended to be a proof of the Thesis. For, an identification of an informal notion with a mathematically precise one is not a subject of a mathematical proof.

Remark. Although the logicians involved in these investigations in the 1930s believed that such a Thesis cannot be a subject of a mathematical proof, more recently, since the 1980s, some scholars claimed that it is provable or has already been proved. These papers and approaches are surveyed and critiqued in the next section.

Let us now return to §7 of Church's (1936) where he provides "positive justification" for the Thesis. There he examines the calculation of a function in a logic, generalizing Gödel's considerations for equational calculi. According to Church, calculating the number theoretic function F in a logic L means the following. Assuming L contains expressions $1, 2, 3, ...$ for positive integers, the identity symbol $=$, and a symbol $\{\ \}(\ )$ for the application of a function of one integer to its argument and L has to satisfy certain conditions which amount to requiring L's 'theorem' predicate to be recursively enumerable. Then, F is considered to be "*calculable within* the logic if there exists an expression f in the logic such that $\{f\}(\mu) = \nu$ is a theorem when and only when $F(m) = n$ is true, μ and ν being the expressions which stand for the positive integers m and n." (p. 357) If a logic L meets the above conditions, then it can be proven that F is a recursive function. Using Kleene's technique in the proof of the equivalence of "general recursiveness" and "μ-recursiveness" in (Kleene 1936).

That is, Church requires every step of the calculation of the function to be recursive, a requirement Sieg refers to as "Church's central thesis." In turn, this central thesis guarantees the function itself to be recursive, and makes Church's argument "subtly circular." Church himself was aware of this weakness of his argument. On the previous page he examines the notion of "algorithm" and at the same point of his analysis, where each step of the algorithm is required to be recursive, he remarks: "If this interpretation or some similar one is not allowed, it is difficult to see how the notion of an algorithm can be given any exact meaning at all." (p. 357, footnote 19)

We see that by 1936 Church used the notion of recursion to define effective calculability instead of λ-definability, although their equivalence was already established. Now we take a quick look at Hilbert and Bernays' *reckonable* functions, deeply influenced by the work of Church, Kleene, and Rosser.[12]

1.4 Hilbert and Bernays' Reckonable Functions

The second supplement of Hilbert and Bernays' *Grundlagen der Mathematik II* (1939, pp. 406-437) is devoted to the notion of calculable function and to Church's undecidability result. Their aim is to detach the notion of calculable number theoretic function from any particular features of formalisms, similarly to Gödel. They call a function *reckonable* ("regelrecht auswertbar") if it is calculable in some deductive formalism (logic), where these formalisms have to satisfy three recursiveness conditions. The main one of these requires that the proof predicate of the formalism has to be primitive recursive, i.e. the theorems of the deductive formalism are recursively enumerable. This requirement is an analogue of what Sieg calls Church's central thesis. Similarly to the case of Church's, if a deductive formalism meets the recursiveness conditions, then it can be proved that the reckonable functions are recursive. As Sieg points out, "this analysis [of Hilbert and Bernays] does *not* overcome the major stumbling block", that is, the semi-circularity of the arguments supporting the formal characterization of mechanical procedure by Gödel and Church, but "it puts it rather in plain view." (2009, p. 565) Interestingly, Hilbert and Bernays do not even mention the work of Turing.[13]

* * *

Thus we see that Gödel, Church, as well as Hilbert and Bernays attempted to characterize mechanical procedures or (effective) calculability of number theoretic functions in terms of restricted calculations within deductive formalisms. As the "elementariness" of the calculation steps were stipulated only by fiat and the arguments that were provided for these characterizations all contained subtle circularities, they were not seen as entirely convincing ones. Kleene in his (1952) lists positive evidence for Church's Thesis available at the time. Under the heading of "heuristic evidence" Kleene emphasizes that: "Every particular effectively calculable function [...] has proved to be general recursive". Furthermore, that "[t]he methods for showing effectively calculable functions to be general recursive" were so developed and routine that its extremely unlikely that "one could describe an effective process [...] which could not be transformed into a general recursive definition". Finally, he mentions that no method was found that would provide an effectively calculable function that is not general recursive. (pp. 319-320) Besides this semi-empirical evidence the "equivalence

[12] On this connection and correspondence between Church and Bernays see Sieg's (1997).
[13] Or Post for that matter.

of diverse formulations" is taken as a strong argument. That is, that all characterizations of effective calculability turned out to be equivalent to each other. Kleene's normal form theorem (1936) was essential in developing these equivalences. The remaining two arguments Kleene mentions in support of Church's Thesis are the step by step argument Church provided in his (1936) and "Turing's concept of a computing machine." As was already noted, it was Turing's[14] different approach and analysis that was seen as a convincing argument. In the next section the concept of computability is examined, starting with Turing's analysis, followed by Gandy's and Sieg's contributions.

[14] And to some extent Post's.

2 Mechanical Computability

In the previous section we examined the work of Gödel, Church, Kleene, and Hilbert and Bernays on the notion of calculable number theoretic function during the 1930s. Their work lead to equivalent characterizations of the notion in terms of recursive functions, λ-definability, and reckonable functions. The shared approach was to investigate the procedure of evaluating a function within a calculus. Both Gödel and Church argued for the correctness of their formal characterization, but their arguments were not viewed, even by themselves, as convincing ones. It was Turing's quite different and original approach in his analysis of computing by a human computer that provided a convincing argument for an equivalent formalization of the notion.[15] In this section Turing's work in his classic (1936) is discussed. It is then followed with Gandy's and Sieg's work building on Turing's.

2.1 Turing's Notion of Computability

In his (1936), *On Computable Numbers, with an Application to the Entscheidungsproblem*, Turing investigates the notion 'computable number', but remarks that "it is almost equally easy to define and investigate computable functions, [...] computable predicates, and so forth." (p. 230) That is, the analysis is clearly applicable to the notion investigated by Church, Gödel, and the others.

Turing introduces the mathematically precise notion of computing machines, which are since called Turing machines. This notion is then used to define computable numbers:

> According to my definition, a number is computable if its decimal can be written by a machine. (p. 230)

Then the informal notion "all numbers which could naturally be regarded as computable" is identified with the mathematically precise notion of computable number. This identification is an analogue of Church's Thesis, where the informal notion of "numbers which could naturally be regarded as computable" stands for the same intuitive notion as effective calculability stood for Church. As Turing machine computability is provably equivalent to λ-definability, it is sometimes referred to as 'Turing's Thesis', while 'Church-Turing Thesis' is frequently used to refer to both or one of them simultaneously.

Turing argues for his Thesis in §9 *The extent of the computable numbers*. His aim is to show that the operations of a human computer, while computing a number, can be mimicked by the machines he defined. Similar to Church, he claims that "all arguments which can be given are bound to be, fundamentally, appeals to intuition" and do not and cannot constitute a mathematical proof of

[15] Although Church in his review of Turing's (1936) seems to be convinced by Turing's analysis, it is clear that he misunderstood it. Gödel referred to Turing's work already by the end of the 1930s, but gained a deeper understanding of it only by the 1950s. As it was mentioned at the end of the previous section, Hilbert and Bernays did not even mention his work in their (1939).

the Thesis. Also similar to Gödel and Church, for Turing "[t]he real question at issue is: 'What are the possible processes which can be carried out in computing a number?"' (p. 249) However, Turing was interested first and foremost in the decision problem, which led him to analyze mechanical computations directly, not through calculable number theoretic functions.

Through the analysis of the human computer Turing's goal is to "split up" the processes "into 'simple operations' which are so elementary that it is not easy to imagine them further divided." (p. 250) As a consequence of the limitations of human cognitive capacities the simple steps are shown to be indeed elementary. Turing's analysis is not concerned with calculations "in a calculus" and the steps allowed during a calculation are not motivated by mathematical practice as in the case of Gödel for example. Rather, what was one step for Gödel while calculating a function, e.g. substituting equals for equals, will be "split up" into further operations by Turing. Let us now turn to the details of Turing's analysis of the human computer computing a number.

At the beginning of the analysis Turing asserts that "[c]omputing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares like a child's arithmetic book." However, as Turing states without any argument, the two-dimensional character of the paper is not essential and "is always avoidable". (p. 249) Thus, calculations are made on a (possibly) two-way infinite one-dimensional tape, divided into squares.

The next, minor, consequence of limitations of human cognitive capacities is that only finitely many symbols are allowed in the alphabet, because "[i]f we were to allow an infinity of symbols, then there would be symbols differing to an arbitrary small extent." (p. 249) As Turing's aim is to "split up" the process of computing into operations for which it is "not easy to imagine them further divided", the symbols the computer is operating on have to be observable "at one glance." (p. 250) Indeed, if infinitely many different symbols were allowed in a unit square, there would be symbols that cannot be distinguished at a glance because of their similarity. That is, this restriction is a consequence of the sensory limitations of the human computer. At the same time, as Turing emphasizes, it is only a minor restriction, as "[i]t is always possible to use sequences of symbols in the place of single symbols." (p. 249)

At any moment during the computation the behavior of the human computer is determined by their "state of mind" and the currently observed squares. Another consequence of the sensory limitations of human computers is that there is a limit B on the number of squares that can be observed at a time. It is, again, not a serious restriction, as the observation of more than B squares can be broken down into successive observations. According to Turing, the limitation on the number of states of mind "are of the same character as those which restrict the number of symbols. If we admitted an infinity of states of mind, some of them will be 'arbitrarily close' and will be confused." (p. 250) Hence, it is a consequence based on cognitive capacities once again. Many scholars found this to be the weakest part of Turing's analysis.[16] It is somewhat surprising,

[16]For example Post (1965), Gödel (1972a) and Kreisel (1972), for the latter, see the Ap-

as a couple pages later Turing dispenses with the state of mind. For, the introduction of the "state of mind" can be avoided "by considering a more physical and definite counterpart of it. It is always possible for the computer to break off from his work, [...] and later to come back and go on with it. If he does this he must leave a note of instructions explaining how the work is to be continued. This note is the counterpart of the 'state of mind'." (p. 253)

In the next step Turing describes the possible operations that the computer can perform. These include changing the state of mind of the computer, changing the content of the tape, or changing the "distribution of observed squares." When it comes to changing symbols, i.e. writing on the tape, "[w]e may suppose that in a simple operation not more than one symbol is altered. Any other changes can be split up into simple changes of this kind. [...] We may, therefore, [...] assume that the squares whose symbols are changed are always 'observed' squares." (p. 250) When the computer shifts its focus, i.e. changes the "distribution of observed squares", another restriction is imposed, namely that "each of the new observed squares is within L squares of an immediately observed square."

We see that Turing ended up with two important conditions, first isolated in Sieg's (1994), based on the sensory limitations of the human computer:

Boundedness: there is a bound on the configurations a human computer can recognize immediately.

Locality: the human computer can change only currently observed squares, and can shift their attention within a bounded distance from the currently observed squares to another ones to be immediately recognizable.

In the boundedness criterion the bound on the configurations is a joint consequence of the limitations on the number of symbols and currently observed squares.

According to Turing "[w]e may now construct a machine to do the work of this computer", i.e. a machine can "mimick" the human computer. Then, in turn, everything that can be computed by these machines can also be computed by Turing Machines as well. As Sieg emphasized in his (1994), this claim can be turned into a precise mathematical theorem:

> *Any number that can be computed by a human computer acting deterministically satisfying the Boundedness and Locality conditions, can be computed by a Turing Machine.*

Where "deterministically" captures the requirement that "[t]he behaviour of the computer at any moment is determined by the symbols which he is observing, and his 'state of mind' at the moment." (p. 250) Reformulating the above theorem for number number theoretic functions, Sieg obtains what he calls Turing's Theorem: number theoretic functions computable by human computers

pendix.

satisfying the above conditions are Turing computable. (1994, p. 94) This theorem is analogous to the theorems of Church, and Hilbert and Bernays

Thus, we see how different Turing's approach is. He is relying neither on the mathematically precise restrictions on process of calculation "in a logic" or some kind of formalism nor on mathematical restrictions implicit in mathematical practice. Instead, he focuses on a human computer computing a number with a pencil on paper, and analyzes this process in terms of the clearly articulated limitations of the cognitive capacities of the computer. This analysis is now seen by most scholars as the most convincing argument for the Church-Turing Thesis, besides the empirical evidence gained since 1936.

Indeed, in 1937 Church reviewed Turing's (1936) and said that compared to general recursiveness and λ-definability, Turing machines have "the advantage of making the identification with effectiveness in the ordinary sense evident immediately" (p. 43).[17] As we saw, Gödel also became convinced by Turing's analysis, but only later. In 1964, Gödel wrote a Postscriptum to his Princeton lectures from 1934. There Gödel asserts that "due to A. M. Turing's work, a precise and unquestionably adequate definition of the general concept of formal system can now be given" (1964, p. 369). Turing's work is described as follows:

> Turing's work gives an analysis of the concept of "mechanical procedure" (alias "algorithm" or "computation procedure" or "finite combinatorial procedure"). This concept is shown to be equivalent with that of a "Turing machine". A formal system can simply be defined to be any mechanical procedure for producing formulas, called provable formulas. (pp. 369-370)

However, it is nowhere indicated by Gödel where such an equivalence was "shown" by Turing.

In the next subsections we take a look at Gandy's work on machine computation and Sieg's structural axiomatization of computability. Gandy's analysis of (parallel) machine computability follows closely Turing's analysis, using physical principles to restrict computation processes instead cognitive limitations of human computers. Sieg's structural axiomatization builds on both Turing's and Gandy's work and generalizes them in an abstract framework.

Remark. In 1936, independently of Turing, Emil Post proposed an almost identical computational model in his *Finite Combinatory Processes, Formulation 1* (1936). Indeed, Turing in his (1950) refers to "logical computing machines introduced by Post and the author [i.e. himself]" (p. 491). Post's (1936) is rather short and does not contain further results, such as undecidability, besides the proposed model of computing. However, it was the result of his work from the 1920s, when he anticipated the incompleteness and undecidability results as well. Post's approach was also rather similar to Turing's, analyzing combinatory processes of finite configurations or strings, not number theoretic functions. Unfortunately Post's work of the 1920s got published only posthumously by his

[17] Though Church did misunderstand Turing's argument in §9.

student and friend Martin Davis as (Post 1965). For detailed analyses and comparisons of Turing's and Post's work see Davis and Sieg's (2015) and Sieg, Szabó and McLaughlin's (2017).

2.2 Gandy's Principles for Mechanisms

Gandy in his (1980) provided principles for mechanical computing devices. He argued that "Turing's analysis of computation by a human being does not apply directly to mechanical devices" as "there are crucial steps in Turing's analysis where he appeals to the fact that the calculation is being carried out by a human being. One such appeal is used to justify the assumption that the calculation proceeds as a sequence of elementary steps. A human being can only write one symbol at a time. But, if we abstract from practical limitations, we can conceive of a machine which prints an arbitrary number of symbols simultaneously." (pp. 123-125) Hence an analysis of "discrete deterministic machines" is needed, where discrete deterministic machines include machines that compute in parallel as well. Gandy's (1980) is dedicated to giving a set theoretic description of such machines directly, which are now called Gandy Machines.

Gandy's analysis of machine computation parallels Turing's analysis of human computation. Where Turing took into account the sensory limitations of humans, Gandy derives his restrictions from two physical constraints. The first such limitation is "that there is a lower bound on the linear dimensions of every atomic part of the device". (p. 126) The second is "the finite velocity of propagation of effects and signals: contemporary physics rejects the possibility of instantaneous action at a distance." (p. 135) Then Gandy advocates four principles that have to be satisfied by "discrete deterministic mechanical devices". First, the computation proceeds in discrete steps, where the states of the machines and the transition function among states can be described in terms of hereditarily finite sets. Second, the set theoretic rank of the states is bounded. Third, the Principle of Unique Assembly, which asserts that any state can be assembled uniquely from parts of limited size. And, finally, the Principle of Local Causation, which requires that the successive state of a computation can be assembled from finitely many kinds of bounded, overlapping regions of the previous state after having been operated on. The restriction to finitely many kinds of bounded regions is taken as the analogue of Turing's boundedness condition of there being a fixed bound on immediately recognizable configurations. It is justified, according to Gandy, by the finite velocity of signal propagation.[18]

Gandy mentions in the introduction a second physical limitation but his analysis actually does not use the other physical limitation, i.e. a lower bound

[18] Gandy presented his machines in an extremely complex way in his (1980). A more accessible and simplified presentation of them can be found in Sieg and Byrnes' (1999). Since Gandy's machines served as part of the motivation for Sieg's axiomatization of Computable Dynamical Systems, I omit a lengthy description of Gandy's machines and direct the reader to the next subsection where Sieg's axiomatization is explained in detail. Sieg's presentation is significantly simpler and he was also able to dispense with Gandy's second principle on the restriction of the set theoretic rank of the states and rely on purely physically motivated restrictive conditions.

on atomic dimensions.[19] As it is assumed that the machines always occupy a finite (but not bounded) region of space, the lower bound on atomic size entails that the machine always consists only of finitely many atoms. This provides justification for the use of hereditarily finite sets.

Gandy also follows Turing's analysis closely in that he claims that his conceptual analysis and mathematical work provides an argument for a thesis, namely for:

> **Thesis M.** *What can be calculated by a machine is computable* [by a Turing machine]. (p. 124)

The argument is provided in two main steps. The careful conceptual analysis supports a more "definite" version of Thesis M:

> **Thesis P.** *A discrete deterministic mechanical device satisfies principles I-IV below.* (p. 126)

Which is followed by a representation theorem:

> **Theorem.** *What can be calculated by a device satisfying principles I-IV is computable* [by a Turing machine]. (p. 126)

This parallelism brings out how closely Gandy followed Turing's analysis.

Not surprisingly, the other direction of the theorem holds as well, thus Turing machines are computable by Gandy machines. Gandy also showed that Conway's Game of Life is computable by his machines. It was important for him, as he viewed the Game of Life as an exemplar of a system that Turing's analysis of computability did not cover, but should be shown to be computable by machines.

Gandy's work was an important motivation of Sieg's axiomatic approach. His axiomatic framework generalizes both Turing's and Gandy's ideas. An important difference is that Sieg gives a structural axiomatization of computability. Thus, there is no need for theses or principles. It means that the emphasis is shifted and it can be treated in the same way as other mathematical axiomatic descriptions. However, it does not mean that an adequate conceptual analysis can be dispensed with, quite the contrary, or that Church's Thesis will become provable. We turn to Sieg's axiomatization in the next section.

2.3 Sieg's Structural Axiomatization of Computability

Here I discuss Sieg's structural axiomatization of computability in detail. Sieg's analysis of the concept builds on the work of both Turing and Gandy. As the result of his analysis he provides an axiomatic characterization of computability in terms of Computable Dynamical Systems. It is important to emphasize that the axiomatically introduced notion of Computable Dynamical System is not yet another theoretical computing device, but an umbrella notion other computing

[19] Gandy's paper was written in a rush in order to meet the submission deadline for the proceedings of the Kleene Symposium where the paper was published.

devices fall under. Thus, the structural axiomatization of computability plays the same role as the structural axiomatic treatment of the notion of group or field: "These abstract notions are distilled from mathematical practice for the purpose of comprehending complex connections, of making analogies precise and of obtaining a more profound understanding." (Sieg 2009, p. 600) Furthermore, Sieg proves a representation theorem, showing that any Computable Dynamical System can be "mimicked" by a Turing machine.

Thus, Sieg's structural axiomatization provides a delimitative unification of the notion of computability. It is a unification as it incorporates both human and machine computation. And it is delimitative, as this structural characterization with clearly stated axioms makes it clear that any computational device that is said to be computationally more powerful than Turing machines has to violate one of the axioms.

Now I turn to the mathematical details of Sieg's axiomatization. Sieg presented it in his *Calculations by Man and Machine: Mathematical Presentation* (2002). It was reprinted as 5.1 and 5.2 of his *On Computability* (2009). Here I survey his axiomatization; as Sieg's (2009) was already referred to in the previous sections, I give the places of the references and quotations from his (2009). I follow those sections closely, exact formulas are taken over for example. I emphasize that I do not claim any originality in this section. Although 5.2 is entitled *Gandy Machines*, they will be called here *Computable Dynamical Systems*. The purpose of the distinctions is twofold. In this way 'Gandy Machines' will refer to Gandy's and De Pisapia's work, leading to a clearer naming convention. But more importantly, this also separates the significantly different methodological approaches of Gandy and Sieg.

Sieg follows closely Turing's argument in his (1936) and Gandy's parallel argument from his (1980) and formulates the following conditions for Turing computors: (i) they operate deterministically on finite configurations; (ii) they recognize in each configuration exactly one pattern (from a bounded number of different kinds of such); (iii) they operate locally on the recognized patterns; (iv) they assemble the next configuration from the original one and the result of the local operation. However, as the goal is to cover parallel computing as well, (ii), (iii) and (iv) have to be slightly modified. In case of (ii), instead of requiring to recognize exactly one pattern, we only require to recognize patterns unambiguously. As to (iii) and (iv), the only change is that operations on all recognized patterns happen at the same time and the next state is assembled from the original state and the results of the local operations.

Formally, we consider pairs $\langle D, F\rangle$ where D is a class of states and F is an operation from D to D that transforms a given state into the next one. States are finite objects and are represented by non-empty hereditarily finite sets over an infinite set U of atoms. Such sets reflect states of computing devices just as other mathematical structures represent states of nature, but this reflection is done somewhat indirectly, as only the $\in$-relation is available. To free the framework from particular representations, *structural classes* S are considered, i.e. classes of states that are closed under $\in$-isomorphisms, instead of specific states. This raises immediately the question, what invariance properties

the state transforming operations F should have or how the F-images of $\in$-isomorphic states are related.

Every $\in$-isomorphism between two states is a unique extension of a permutation π of the atoms of those states. We can use these permutations to set the law-like connections between structural classes S via *structural operations* G. As the atoms of U are used to represent parts of the computing machines, such as the tape of a Turing Machine for example, we would like to use the same atoms throughout the computation. Thus we require the structural operations, on any state x, that $G(\pi(x))$ and $\pi(G(x))$ not only have to be $\in$-isomorphic, but the isomorphism has to be the identity on the atoms in the support of $\pi(x)$. We will say that $G(\pi(x))$ and $\pi(G(x))$ are *$\in$-isomorphic over* $\pi(x)$ and will write $G(\pi(x)) \cong_{\pi(x)} \pi(G(x))$. The following figure depicts this connection:

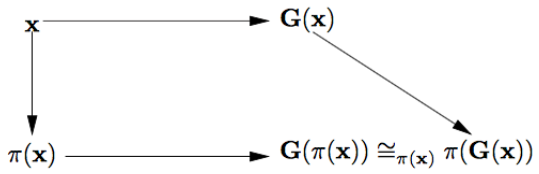


Figure 1, from (Sieg, 2009, p. 588)

So far we addressed only (i) from the above list, now we turn to *patterns* and *local* operations. If x is a given state, regions of the next state are determined *locally* from particular *parts for* x on which the computor can operate. *Boundedness* requires that there is only a bounded number of different kinds of parts, i.e. each part lies in one of a finite number of isomorphism types, or, using Gandy's terminology, *stereotypes*. So let T be a fixed finite class of stereotypes. A part for state x that is a member of a stereotype T is called a *T-part for x*. A T-part y for x is a *causal neighborhood for* x given by T if there is no T-part y^* for x such that y is $\in$-embeddable into y^*. In notation $y \in Cn(x)$. The causal neighborhoods for x will also be called *patterns in* x. Finally, the local changes will be effected by structural operations that work on causal neighborhoods. Having also given points (ii) and (iii) a proper mathematical explication, the assembly of the next state has to be determined.

The values of the structural operations are in general not exactly what we need in order to assemble the next state, because the configurations may have to be expanded or different local operations may overlap. This involves the addition and coordination of new atoms. To address that issue we introduce *determined regions* $Dr(z, x)$ of a state z; they are $\in$-isomorphic to $G(y)$ for some causal neighborhood y for x and must satisfy a technical condition on the "newness" of atoms. More precisely, $v \in Dr(z, x)$ if and only if $v <^* z$[20] and there is a $y \in Cn(x)$, such that $G(y) \cong_y v$ and $Sup(v) \cap Sup(x) \subseteq Sup(y)$. The last condition for Dr guarantees that new atoms in $G(y)$ correspond to new atoms in v, and that the new atoms in v are new for x. Now, with the

[20]Where $y <^* x$ if $y \neq x$ and y has the same root as x and its leaves are also leaves of x.

use of structural operation G and determined region Dr we can coordinate the addition of new atoms in case of particular causal neighborhoods.

However, if the outcome of the structural operation on two or more causal neighborhoods overlap, we can quickly run into further complications. These issues can be resolved by a second local operation and a second set of stereotypes. Causal neighborhoods of type 1 are parts of neighborhoods of type 2 and the overlapping determined regions of type 1 must be parts of determined regions of type 2, so that they fit together appropriately. The next state of the computable discrete dynamical system is assembled by simple set theoretic operations. (There is another, rather harmless case of overlapping causal neighborhoods. That is, when one causal neighborhood contains the causal neighborhood of another, applicable rule. In this case we adopt the convention that it is always the rule with the largest causal neighborhood that is applied.)

We say that $M = \langle S; T_1, G_1, T_2, G_2 \rangle$ is a *computable discrete dynamical system* on S, where S is a structural class, T_i a finite set of stereotypes, G_i a structural operation on $\cup T_i$ (i =1 or 2), if and only if for every $x \in S$ there is a $z \in S$ such that satisfies the following axioms:

(L.1) $(\forall y \in Cn_1(x))(\exists! v \in Dr_1(z,x))v \cong_x G_1(y)$;
(L.2) $(\forall y \in Cn_2(x))(\exists v \in Dr_2(z,x))v \cong_x G_2(y)$
(A.1) $(\forall C)[(C \subseteq Dr_1(z,x))\& \cap \{Sup(v) \cap A(z,x) \mid v \in C\} \neq \emptyset \rightarrow$

$$(\exists w \in Dr_2(z,x))(\forall v \in C)v <^* w];$$

(A.2) $z = \cup Dr_1(z,x)$.

where L stands for Locality and A for Assembly and $\exists!$ is the existential quantifier expressing uniqueness. The condition $\cap \{Sup(v) \cap A(z,x) \mid v \in C\} \neq \emptyset$ in A.1 expresses that the determined regions v in C have new atoms in common, i.e. they *overlap*.[21]

The next state z is determined up to $\in$-isomorphism over x. A *computation by* M is a finite sequence of steps via the structural operations, and it halts when the operations on a state w yield w as the next state.

A function **F** will be called computable if and only if there is a Computable Discrete Dynamical System whose computation determines – under a suitable encoding and decoding – the values of **F** for any of its arguments.

About the computational power of Computable Dynamical Systems Sieg shows that Computable Dynamical Systems are computationally reducible to Turing Machine, which leads to the following:

[21] These axioms cover both the sequential and the parallel case. Axioms for only the sequential case, which Sieg presents before the general case, are considerably simpler as there is no need for secondary causal neighborhoods and determined regions. For, there is no overlap of new atoms even when expanding, because there is always only one causal neighborhood that is present. For future reference I include here the axioms for the sequential case:
(L.1) $(\exists! y)\ y \in Cn(x)$,
(L.2) $(\exists! v \in Dr(z,x)) \cong_x G(Cn(x))$,
(A.1) $z = (x \setminus Cn(x)) \cup Dr(z,x)$.

Representation Theorem. *Any particular Computable Dynamical System is computationally reducible to a Turing Machine.*

As Turing Machines are also Computable Dynamical Systems (see the next subsection for details).

In 1990, Lamport and Lynch in their survey article about distributed computing made the following "important" and "informal observations" about the not yet established foundational concepts of parallel and distributed computing:

> Underlying almost all models of concurrent systems is the assumption that an execution consists of a set of discrete events, each affecting only part of the system's state. Events are grouped into processes, each process being a more or less completely sequenced set of events sharing some common locality in terms of what part of the state they affect. For a collection of autonomous processes to act as a coherent system, the process must be synchronized. (Lamport and Lynch, 1990)

It is clear that these informal observations line up well with the informal ideas behind Sieg's criteria for parallel computing. Namely, the principle of Local Causation concurring with "events each affecting only part of the system's state". The requirement of deterministic operation on the causal neighborhoods with the "more or less completely sequenced set of events sharing some common locality". Finally, the principle of Global Assembly corresponding more or less with synchronization.

A Small Change in the Presentation

Below a slightly modified version of Sieg's framework will be used. The small change leads to an equivalent framework. The reason for this modification is that Gandy's and Sieg's approach was primarily motivated to accommodate Conway's Game of Life, a parallel model which can grow without boundaries and every part of it is recomputed and rebuilt during each step. It is clear that the Game of Life is important for the theory of computability for these properties. However, at the same time, these properties distinguish it from more realistic models of parallel computing.

Indeed, most parallel models do not grow with time. This leads to the effect that there will be no need for secondary causal neighborhoods and determined regions, an issue to which I will come back at the end of the chapter. Perhaps more importantly, standard parallel models of computing do not recompute and rebuild themselves at every computational step. The first small modification of Sieg's axiomatization addresses this issue. Sieg's second assembly axiom (A.2) captures this feature of complete rebuilding:

$$z = \cup Dr_1(z, x).$$

Which makes the axiom different in its structure from the assembly axiom in the sequential case (see footnote 21), where only one piece of the device is changed:

$$z = (x \setminus Cn(x)) \cup Dr(z, x).$$

An axiom that suits the picture of reassembling only parts of the machine that are taking part in the parallel computation in the current step is:

$$z = (x \setminus (\cup Cn_1(x))) \cup (\cup Dr_1(z, x)),$$

which is closer in its form to the sequential case as well. Furthermore, it reduces to the original axiom in case of Conway's Game of Life as $\cup Cn_1(x)$ exhausts x.

This is not a serious modification of the framework. One could include every part of the computing device in a separate rule which has that part both as a causal neighborhood and a determined region. This way every part that is not being modified by the computation itself is being "cut out" and reassembled. The original proofs go through as well, as the things that have to be dealt with are only those things that are being changed during a computational step.

2.4 Previous Results about Computable Dynamical Systems

In this subsection I survey some previous results to show how this axiomatic framework has been used in practice. First I will follow Sieg's (2009) on treating Turing Machines and Conway's Game of Life, and then I give a short description of Artificial Neural Networks as Computable Dynamical Systems based on De Pisapia's (2000).

Turing Machines[22]

Let us consider a Turing Machine with internal states $q_0, ..., q_m$ and symbols $s_0, ..., s_k$. Its program is a finite set of quadruples in the form $q_i s_j c_k q_m$, where q_i is the current state of the machine, s_j is the currently read symbol, c_k is the action to be performed, i.e. print another symbol, move to left L or to right R, and, finally, q_m is its new internal state. The tape of the machine is described with overlapping ordered pairs of atoms:

$$\mathbf{Tp}{:=}\{\langle b, b\rangle, \langle b, c\rangle, ..., \langle d, e\rangle, \langle e, e\rangle\}$$

The leftmost possibly non-blank atom of the tape is c while the rightmost is d. The symbols are represented by $\overline{s_j} := \{a\}^{(j+1)}$, $0 \leqq j \leqq k$; the internal states are given by $\overline{q_j} := \{a\}^{(k+1)+(j+1)}$, $0 \leqq j \leqq l$, where a is a fixed atom. The tape content is given by:

$$\mathbf{Ct}{:=}\{\langle c, \overline{s_{j_0}}\rangle, ..., \langle d, \overline{s_{j_r}}\rangle\}$$

[22]Here I follow closely the pages 596-597 of Sieg's (2009), though the notation is slightly changed.

Then, the instantaneous description is given as the union of **Tp**, **Ct**, and $\langle r, \overline{q_i} \rangle$ with r being a square of **Tp** and q_i being the current internal state. The structural set S of states is obtained as the set of all instantaneous descriptions closed under $\in$-isomorphisms. The causal neighborhoods for program lines starting with $q_i s_j$ on which G operates take the form:

$$\{\langle r, \overline{q_i} \rangle, \langle r, \overline{s_j} \rangle, \langle t, r \rangle, \langle r, u \rangle\}$$

Let us now consider the program line $q_i s_j s_k q_l$ which prints s_k. In this case the structural operation G leads to the following determined region:

$$\{\langle r, \overline{q_l} \rangle, \langle r, \overline{s_k} \rangle, \langle t, r \rangle, \langle r, u \rangle\}.$$

For the program line $q_i s_j R q_l$, where the scanner head is to be moved to the right we have to distinguish two cases. First, when the currently read square r is not the rightmost square on the tape. In this case G simply yields to:

$$\{\langle u, \overline{q_l} \rangle, \langle r, \overline{s_j} \rangle, \langle t, r \rangle, \langle r, u \rangle\};$$

while in the second case G results in:

$$\{\langle *, \overline{q_l} \rangle, \langle r, \overline{s_j} \rangle, \langle *, \overline{s_0} \rangle, \langle t, r \rangle, \langle r, * \rangle, \langle *, u \rangle\};$$

where $*$ is a new atom appended to the end of the tape. Program lines involving moving to the left can be treated similarly. It is easy to verify that a Turing Machine presented as above is a Computable Dynamical System.

Conway's Game of Life

Conway's Game of Life was invented by mathematician John Horton Conway, motivated by von Neumann's work on self-reproducing automata (1966). It was first published by Martin Gardner in his column on recreational mathematics in the Scientific American (Gardner, 1970).

The universe of the game consists of a (possibly) infinite two-dimensional grid of square cells. At each step cells are in one of two states: alive or dead. The rules of birth, death, and staying alive are the following:

- every living cell with 2 or 3 living neighbors stays alive,
- every living cell with only 1 living neighbor dies from isolation,
- every living cell with 4 or more living neighbors dies from overpopulation,
- every dead (empty) cell with exactly 3 living neighbors will be born.

As Gardner emphasizes, "[i]t is important to understand that all births and deaths occur *simultaneously*."

The obvious allusion to living organisms and its richness of possible configurations led to a growing interest which resulted in the independent research area called 'cellular automata' theory. What is important for computational models,

and was already emphasized by Gardner, is that the determination whether a cell will be dead or alive in the next step happens simultaneously for each cell. However, Conway's Game of Life is a rather unusual model of parallel computing. Unusual in the sense that unlike real parallel computing devices, it can grow beyond any limit. It was this feature of the Game of Life that made it important for Gandy. As we will see, it was the Game of Life that required the introduction of secondary operations.[23] Here I follow pages 597-598 of Sieg's (2009) closely.

In this presentation we distinguish between *internal cells*, which are surrounded by a layer of *border cells*; these are jointly called real cells. Furthermore, surrounding real cells there is an additional layer of *virtual cells*. Border and virtual cells are dead by convention. The layering ensures that each real cell is surrounded by a full set of eight neighboring cells.

A real cell a with neighbors $a_1, ..., a_8$ and state $s(a)$ is given by:

$$\{a, s(a), \langle a_1, ..., a_8\rangle\}$$

The neighbors are given in a "canonical" order starting with the square in the leftmost top corner and proceeding clockwise. The state $s(a)$ of an atom a will be $\{a\}$ in case it is alive and $\{\{a\}\}$ if it is not. The T_1 causal neighborhoods of real cells are of the form:

$$\{\{a, s(a), \langle a_1, ..., a_8\rangle\}, \{a_1, s(a_1)\}, ..., \{a_8, s(a_8)\}\}$$

As the structural operations G_1 are rather simple for internal cells, the example shown here will deal with border cells. In the following diagram the v's indicate virtual cells, the b's border cells, the $\{a\}$'s internal cells that are alive. The $*$'s represent new atoms that are added in the next step of the computation. Let's see how it comes about in an example:

$*_0$	$*_1$	$*_2$	$*_3$	$*_4$	$*_5$	$*_6$	$*_7$		
v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	v_{10}
$\{a_0\}$	$\{a_1\}$	$\{a_2\}$	$\{a_3\}$	$\{a_4\}$	$\{a_5\}$	$\{a_6\}$	$\{a_7\}$	b_9	v_{11}

Consider the darkly shaded square b_3 with its neighbors, i.e. its presentation:

$$\{b_3, \{\{b_3\}\}, \langle v_2, ..., b_2\rangle\}$$

applying G_1 to its causal neighborhood yields:

$$\{\{b_3, \{b_3\}, \langle v_2, ..., b_2\rangle\}, \{v_3, \{\{v_3\}\}, \langle *_2, *_3, *_4, v_4, b_4, b_3, b_2, v_2\rangle\}\}$$

[23] Another important feature of the Game of Life is its computational power. In 2002 Paul Chapman constructed a Game of Life initial pattern (http://www.igblan.free-online.co.uk/igblan/ca/) that implements a Universal Minsky Register Machine, which is equivalent to a Universal Turing Machine in computational power. Similar results are available about several different versions of cellular automata.

where $*_2$, $*_3$, and $*_4$ are new atoms and v_3 has been turned from a virtual cell into a real one, namely to a border cell. If we now look at the border cell b_2 we can recognize that G_1 yields a rather similar set. However, we have to ensure, that two of the three new atoms created around b_2 coincide with $*_2$ and $*_3$. That is, the secondary operations have to be used to conduct this unification of new atoms. That is, the T_2 causal neighborhood around b_3 will be the union of the T_1 neighborhoods of b_2, b_3, and b_4. For, its only the immediate neighbors of b_3 whose causal neighborhoods overlap with it. Then G_2 yields the set with shared presentations of the cells v_2, v_3, and v_4. One can verify that this structure is a Computable Dynamical System that computes Conway's Game of Life.

Artificial Neural Networks

The representation of Artificial Neural Nets as Computable Dynamical Systems was worked out by De Pisapia in his (2000). Here I give a rather short summary, only indicating the mathematical solutions without going into any detail.

Artificial Neural Nets are represented as directed graphs. The nodes stand for the neurons and their connections are indicated by directed edges. The network is ordered into layers: an input layer which takes its incoming signals from the external world; output layer, i.e. the set of units whose activation functions reach the external world; one or more hidden layers which are situated between the input and output layers.

In the formal model node j is receiving input from its ancestors $1, ..., n$ with the incoming signals denoted by $x_1, ..., x_n$. Furthermore, each incoming signal is weighted by a factor $w_{i,j}$ (from ancestor i to node j in question). The total incoming signal strength is denoted by net_j and it is the dot product of the vectors $(x_1, ..., x_n)$ and $(w_{1,j}, ..., w_{n,j})$, i.e.:

$$net_j = \sum_{i=1}^{n} x_i \cdot w_{i,j}$$

The output a_j of the neuron j, called activation, is a function of the total incoming signal net_j. The activation function can take several forms. For example, it can be a binary threshold unit, i.e. $a_j = 1$ if net_j reaches a certain threshold and 0 otherwise; it can be a linear unit, i.e. $a_j = k \cdot net_j$ where $k \in \mathbb{R}$; or a sigmoid unit, where $a_j = (1 + e^{-net_j})^{-1}$ and so on.

The learning or training of an Artificial Neural Net happens through modifying its weights $w_{i,j}$ to reach the "desired" input/output behavior. After each input the weights may be changed; the new weight can be written as $w_{i,j} + \triangle w_{i,j}$ for each edge. $\triangle w_{i,j}$ is a function of the activation of the nodes i and j, and thus can be determined locally. Learning rules can take several forms, the following are some simple examples: $\triangle w_{i,j} = \varepsilon a_i a_j$ where $\varepsilon \in \mathbb{Q}$ and is called the learning rate; $\triangle w_{i,j} = \varepsilon(t_j - a_j)a_i$ where t_j is the target activation of node j. The new weights are calculated in parallel for each edge.

In De Pisapia's representation of Artificial Neural Networks the difficulty arose from dealing with the weights, activations, and further real numbers. Instead of using rounded rational approximations of real numbers, De Pisapia uses

computable real numbers to achieve the most general representation. Following Pour-El and Richard's (1989), computable reals are defined with the aid of four sequences of rationals. As a consequence, with each real number four Turing Machines are associated that calculate the four sequences. These Turing Machines are located on the artificial neurons, so the activation and weight updates can be calculated locally. As the activation of a node depends only on the activations of its ancestors and the weight updates of an edge depend only on the activations of the two neurons it connects, all causal neighborhoods are indeed finite in size and can be calculated with in parallel.

It has to be pointed out as it will be relevant later that there is no need for secondary structural neighborhoods and operations (T_2 and G_2) to accommodate Artificial Neural Networks. As the networks are not "growing", i.e. no new atoms are introduced, there is no need to coordinate between primary causal neighborhoods. Sieg emphasized the importance of the question whether these secondary operations play a role in human mental processes:

> "Is there a natural subclass of Turing computable functions in which these neural nets lie?", and "Are there mental processes for whose representation this aspect [i.e. secondary neighborhoods and operations] of Gandy machines might be crucial?". These are important and most appealing questions. (2002, p. 257)

In the next two chapters I will come back to this question: there is no need for secondary neighborhoods or operations when dealing with realistic models of parallel computing. In case of models of cognition, the models treated in detail do not require these features, but such models will be mentioned which most likely would have to use them.

* * *

Many scholars were wishing for an axiomatic treatment of computability with the hope that once such a generally accepted axiomatization is in place, Church's Thesis might become provable. The classical view on this matter is that the Thesis is not provable. There are few scholars who argued for the provability of Church's Thesis, or even that it already has been proven. The next section provides a critical survey of the work of these scholars. As this question is not central for the remainder of the Dissertation, the reader can jump over it and continue with the 4th Section with no loss of understanding of the later chapters.

3 Is Church's Thesis Provable?

Here I give a critical overview of the literature on the provability of Church's Thesis. The arguments claiming the provability of Church's Thesis will be separated into two groups. The first argues for a possible mathematical proof for the Thesis, while the second argues for the possibility of a formal, axiomatic proof. That is, the first group uses a way more broader notion of proof than the second, which approaches computability axiomatically. Before turning to these approaches, we are going to take a look at Church's, Kleene's and others treatment of the thesis who considered it unprovable.

3.1 The Classical View

The generally received view of the Thesis is that it is not provable. In his classical (1936) Church introduced his Thesis as a definition with the following remark:

> We now define the notion, already discussed, of an *effectively calculable* function of positive integers by identifying it with the notion of a recursive function of positive integers. This definition is thought to be justified by the considerations which follow, so far as positive justification can ever be obtained for the selection of a formal definition to correspond to an intuitive notion. (p. 356)

It is implicit from Church's description that he does not think that such an identification can be a subject of a proof. Kleene, who dubbed this identification Church's Thesis, is even more explicit in his Introduction to Metamathematics:

> Since our original notion of effective calculability of a function is a somewhat vague intuitive one, the thesis cannot be proved. (1952, p. 317)

And later in the same section on Church's Thesis he writes:

> While we cannot prove Church's thesis, since its role is to delimit precisely an hitherto vaguely conceived totality, we require evidence that it cannot conflict with the intuitive notion which it is supposed to complete; i.e. we require evidence that every particular function which our intuitive notion would authenticate as effectively calculable is general recursive. The thesis may be considered a *hypothesis* about the intuitive notion of effective calculability, or a mathematical *definition* of effective calculability; in the latter case, the evidence is required to give the theory based on the definition the intended significance. (Italics mine, pp. 318–319)

Indeed, Kleene straightforwardly rejects the possibility of proving Church's Thesis.[24]

[24] Shapiro addresses the "vagueness" of Church's Thesis in the veins of contemporary analytic philosophy in his (2015, pp. 284–285). As it is not relevant for the current issues, we do not enter into its discussion.

Turing thought about this issue rather similarly. In his (1936) he argued for his identification of computable numbers in the intuitive sense with that of numbers computable by his machines. An identification, which is equivalent to Church's Thesis and sometimes is labeled as Turing's Thesis. Turing argued famously for this identification in §9 of his paper. It is this conceptual analysis provided by Turing in that section that convinced several scholars about Church's (or Turing's) Thesis. The introductory lines of §9 make it clear that Turing holds a similar position to Church and Kleene:

> All arguments which can be given are bound to be, fundamentally, appeals to intuition, and for this reason rather unsatisfactory mathematically. (p. 135)

Indeed, Turing claims that when arguing for the identification, one always has to appeal to intuition. And, furthermore, that it will keep one from getting a purely mathematical argument or result. Later in the 1950s Turing described his stance in a bit more detail.

Turing in his expository paper (1954) introduced wider audiences to the undecidability phenomena in a playful yet rigorous way through puzzles. He used some specific, well known puzzles to discuss what their solvability means. Then he introduced the formal notion of a substitution puzzle. The analogue to Church's Thesis is stated in the following way:

> *Given any puzzle we can find a corresponding* substitution puzzle *which is equivalent to it in the sense that given a solution of the one we can easily use it to find a solution of the other.* (p. 15)

Similarly to Church's Thesis, here the intuitive notion of puzzle is identified with the precise notion of substitution puzzle. Turing's comment on the status of the above thesis is rather close to Kleene's:

> This statement is still somewhat lacking in definiteness, and will remain so. [...] The statement is moreover one which one does not attempt to prove. Propaganda is more appropriate to it than a proof, for its status is something between a theorem and a definition. In so far as we know *a priori* what is a puzzle and what is not, the statement is a theorem. In so far as we do not know what puzzles are, the statement is a definition which tells us something about what they are. (Turing 1954, p. 15)

Although Turing seemingly leaves open the possibility to have a theorem here, that would require us to have *a priori* knowledge of what puzzles are. In case of effective calculability, most people believe that we do not have such *a priori* knowledge.

Let us go back to Kleene's treatment of the Thesis. According to him, it is either a hypothesis or a mathematical definition. Considering it as a mathematical definition, he clearly follows his advisor, Church. The notion of hypothesis was introduced to the discussion in Post's (1936) and strongly supported by Kalmár in his (1959). According to Post:

> [T]o mask this identification [i.e. Church's Thesis] under a definition hides the fact that a fundamental discovery in the limitations of the mathematicizing power of Homo Sapiens has been made and blinds us to the need of its continual verification. (1936, p. 105, footnote 8)

Indeed, mathematical definitions in many cases do not require justifications, for they are 'just' definitions. On the other hand it is clear that it is not only Post who required the Thesis to be justified or argued for. We have already seen Church claiming that the Thesis as a "definition is thought to be justified by the considerations which follow, so far as positive justification can ever be obtained for the selection of a formal definition to correspond to an intuitive notion"; Kleene requiring evidence; while Turing mentioning "appropriate propaganda" for it.

Kalmár agreed with Post that it should not be taken as a definition either. For if it would be regarded as a definition, there would be a chance that in the future someone would be able to define a function which is not effectively calculable in Church's sense, yet it could be effectively calculated for any given arguments. Hence Church's Thesis is possibly refutable while definitions are not.

It is clear that the scholars mentioned above all believed that Church's Thesis is a statement that cannot be proved, at least not purely mathematically. They differed in the question whether to call it a 'thesis' or a 'definition', but seemingly only in terms of how to call it. That is, everyone agreed that arguments have to be given for its acceptance (or rejection). From now on this view will be referred to as the classical view.

Thus, the "classical" view of Church's Thesis holds that it is true but unprovable because it involves an informal notion. Some of the formal or axiomatic approaches claim, however, to prove the Thesis. Before turning to the literature on the provability of the Thesis and axiomatic approaches thereof, I will describe what I believe to be the right picture, when it comes to arguments and proofs of the Thesis and further statements surrounding it.

Church's Thesis involves the informal notion of effective calculability, Turing's equivalent Thesis involves the informal notion of computability by a human computer. Of course just because they are informal, it does not mean that they are not subject to rigorous analysis. However, such an analysis has to be a conceptual analysis, involving philosophical, cognitive and possibly psychological arguments as well. One possible goal of such analyses is to lead to well motivated and well argued formal, mathematical notions.

One can prove that every number theoretic function that is computable by Turing Machines is general recursive. Such mathematical results are of genuine interest only if Turing Machines capture the capabilities and limitations of human computers. However, this mathematically precise and provable statement is not Church's Thesis. The proof of the statement involves two mathematically precise notions, in contrast to Church's Thesis. Furthermore, it is relevant for Church's Thesis only in case if the formalization of computability by a human

computer as a Turing Machine is well motivated and convincing.

Thus, one can prove mathematical theorems about the scope of computational models, e.g. that Turing Machines compute exactly the recursive functions. However, it is always a question whether the computational model captures the informal notion or not. And it is a question that is not, or at least not only, a mathematical question. A conceptual analysis is needed in these situations, connecting the informal and the formal notions. It is exactly this connection that takes the role of a Thesis, and is argued for by a conceptual analysis which is not entirely mathematical, and, hence, cannot constitute a purely mathematical proof.

Now let us turn to positions that differ from the classical view. Namely scholars who claim that Church's Thesis is provable at least in principle, or it is already proven.

3.2 Church's Thesis is Provable by a Mathematical Proof in Principle

Contrary to the classical view there are scholars who believe that Church's Thesis is, at least in principle, provable. First we look at those, who believe that there is a mathematical proof of the Thesis, but do not expect it to be a formalized, axiomatic proof (in, say, Zermelo-Fraenkel set theory). As a natural consequence they do not attempt to axiomatize effective calculability or computability. This stance is represented by Mendelson (1990, 2006), Shapiro (1981, 1993, 2006, 2015) and Black (2000). They do not expect to have a formal proof of the Thesis, because they accept that the Thesis involves informal or vague notions.[25]

Mendelson argues that there are several mathematical arguments, and other claims (that could have been called theses but are not) which involve informal notions and are parts of mathematical proofs. Somewhat surprisingly, he claims that in case of Church's Thesis it is not even about the identification of an informal and a formal notion, because:

> the notion of partial-recursive function [is], in an essential way, no less vague and imprecise than the notion of effectively computable function; the former are just more familiar and are part of a respectable theory with connections to other parts of logic and mathematics. (1990, p. 233)

Thus, Church's Thesis is an identification of two, similarly vague and imprecise notions. He adds that even sets are not clearer notions than the ones defined in their terms, for example the notion of function using ordered pairs.

Furthermore, Mendelson states, "a straightforward argument can be given for" the "easier half" of Church's Thesis:

[25] Surprisingly, they never consider the notion of effective calculability as an open ended notion, which could change their arguments. Shapiro in his (2006) discusses effective calculability in terms of "open texture," but his assessment is that although it is currently an "open texture" notion, at some point it might stop being one.

> (The so-called initial functions are clearly effectively computable; we can describe simple procedures to compute them. Moreover, the operations of substitution and recursion and the least-number operator lead from effectively computable functions to effectively computable functions.) This simple argument is as clear a proof as I have seen in mathematics, and it is a proof in spite of the fact that it involves the intuitive notion of effective computability. (1990, p. 233)

Even though many people would find the above argument convincing, many would not accept it as a proof. Mendelson addresses the question of the notion of proof used here only in passing:

> The fact that it is not a proof in ZF or some other axiomatic system is no drawback; it just shows that there is more to mathematics than appears in ZF. (1990, p. 233)

Thus, it is clear that the notion of proof is taken to be broader here than simply formal deductions in an axiomatic system. We will return to the understanding of the notion of proof below.

Interestingly, although Mendelson lists several other theses[26] from mathematics to show that Church's Thesis does not have such a special or unique status, he never claims that any of the other theses have been proved or are provable, only that they are "well-accepted". It is only Black who claims that one similar thesis has already been proved, namely, what he calls "Dedekind's thesis." That thesis is the claim, that "the second-order Peano axioms capture the structure of the intuitive natural number system." According to him "[w]e do not call this a thesis, because it is *proved* in both directions." (Black 2000, p. 250) But he only provides the following sentence as the alleged proof of the thesis:

> It is clear that the natural numbers satisfy the Peano axioms, and by the categoricity of those axioms any system satisfying them must be isomorphic with the natural numbers. (Black 2000, p. 250)

Again, this sentence is not accepted as a proof by many. Even if it would be, the notion of proof has to be way more inclusive, than allowing only formal deductions in an axiomatic system, for the above sentence to be considered as a proof. For, one has to recognize that the word "clear" in Black's argument carries the whole argument without actually being clear about what it amounts to.

Shapiro, who also believes that Church's Thesis is provable in principle, attempts to give some hints about what that broader notion of proof could possibly be. He gives a "definition" of what a proof is in his (1993):

[26] Among others, such as the definition of function in terms of sets and ordered pairs or the $\varepsilon - \delta$ definition of continuity.

> For present purposes, let us define a "proof" to be a rationally compelling argument, one that a mathematician (*qua* mathematician) should find thoroughly convincing. (p. 69)

This "definition" is rather vague. Does a "rationally compelling argument" always count as a mathematical proof? Because the question is not whether we have good reasons to believe that Church's Thesis is true, the question is whether a mathematical proof of it can be given. Is it the "*qua* mathematician" addendum that is supposed to ensure that the proof is mathematical? If it is, then we are short of an explanation of what it means to be convinced for "a mathematician (*qua* mathematician)." Furthermore, what does "thoroughly convincing" amount to? We certainly do not call every convincing argument a proof; is "thoroughly" supposed to establish that the argument is more than just convincing, so much so that it can be considered as a proof?

Upon providing the broad "definition" of proof, Shapiro also addresses the formalizability of an assumed mathematical proof of Church's Thesis, i.e. how such a proof would relate to the more restricted version:

> No doubt, the study of computability in Turing (1936), or anywhere else for that matter, could be cast in a formal deductive system or in ZF. But that would not be the end of the matter. The issue would then be to determine that the resulting derivation is a good "translation" of the informal arguments. Can *that* be established with a formal proof or a ZF-proof? (1993, p. 67)

Thus he believes that if a mathematical proof can be given, a question would remain about the adequacy of its formalization, which cannot be settled formally. Which leads him to the statement:

> The conclusion so far, is that if there is to be a mathematical proof of Church's Thesis, the proof cannot be fully captured with a formal proof or a ZF-proof. (1993, p. 68)

It not only means that if Church's Thesis has a mathematical proof it can only fit a broader notion proof. Shapiro seems to admit that if there is a proof of the Thesis, it will be *in principle* unformalizable.

In a later paper Shapiro characterizes the situation as follows:

> There would be an intuitive or perhaps philosophical residue connecting the adequacy of the formal or ZFC surrogates to the original, pretheoretic notion of computability. (2015, p. 291)

Namely, that the argumentation for the adequacy "perhaps" involves philosophical argumentation. Shapiro seems to give a coherent picture, the only thing that is obscure at this point is that how can an argument, that involves philosophical work and is unformalizable in principle be considered as a mathematical proof? Compared with the picture I provided in the previous section, the only difference is Shapiro's insistence to call the arguments for the adequacy of the

Thesis mathematical proofs, but he agrees that there always be some "intuitive or perhaps philosophical residue" concerning these issues.

In the literature considering Church's Thesis the classical view, i.e. that the Thesis is true but unprovable, was recently defended by Folina (1998, 2006). She agrees that there is more to mathematics besides formal proofs in axiomatic systems. However, she emphasizes that usually only such arguments are considered to be mathematical proofs, that are thought to be formalizable, at least in principle. That is, of course not every proof that is accepted as a proof is a gapless proof in a formal axiomatic system, but the community believes that they can be turned, at least in principle, into such proofs. Folina claims that if Mendelson and Shapiro want unformalizable arguments to be accepted as proofs, they "need to show [...] that there can be proofs independent of *any* axiomatic system." (1998, p. 312) And so far they have not done so. To add to Folina's point, it seems that their sole motivation to include such unformalizable arguments among proofs is to be able to say that Church's Thesis is provable. That is, they have no independent source or motivation for reconsidering the notion of mathematical proof, which after the revision includes arguments for Church's Thesis as mathematical proofs as well.

Folina diagnoses the source of the problem with Mendelson's and Shapiro's argument as follows: there are two separate questions concerning the Thesis; namely, (a) whether it is provable and (b) whether it is mathematical. According to her, both Mendelson and Shapiro want to answer 'yes' to both questions. But the way they intertwine the two questions leads to a false dilemma:

> The way their discussions intertwine the two questions, however, implies that rejecting Church's Thesis as provable is tantamount to rejecting it as part of mathematics. (1998, p. 313)

It is a false dilemma, as it can be maintained, as in the classical view, that Church's Thesis is a mathematical thesis that is not provable. As Folina emphasizes: "to claim that something is not mathematically provable is *not* to claim that it is not part of mathematics." (p. 313)

Gandy on Church's Thesis

Gandy, uniquely in the literature, claims that Church's Thesis already has been proven. More precisely, he claims that Church proposed and gave arguments for his Thesis, but it was Turing who proved an analogous theorem. That is, Gandy claims that Turing's *analysis* proves[27] the following statement:

> *Any function which can be calculated by a human being can be computed by a Turing machine.* (Gandy 1988, p. 82)

Which is later rephrased and stated in a slightly more exact form as "10.3.1. Theorem":

[27]Sometimes Gandy calls it a proof sketch, or a proof that contains gaps that can be easily filled, just as in case of other mathematical proofs.

> *Any function which can be calculated by a human being following a fixed routine is computable.* (Gandy 1988, p. 83)

Although Turing's analysis in §9 of his (1936) is viewed as the most convincing argument for the Thesis, it is not claimed to be a proof by anyone but Gandy. Church in his review of Turing's (1936) writes that it is Turing's analysis that "has the advantage of making the identification with effectiveness in the ordinary (not explicitly defined) sense evident immediately" (1937, p. 43), but does not take it to be a proof. Why does Gandy believe that Turing proved the above theorem? Turing himself remarked about his analysis:

> All arguments which can be given are bound to be, fundamentally, appeals to intuition, and for this reason rather unsatisfactory mathematically. (1936, p. 135)

To understand Gandy's claims we first have to go back to his interpretation of Church's work and the evidence Church provided for the Thesis. Gandy lists the usual kinds of evidence, such as the confluence of ideas, i.e. the equivalence of different precise notions that capture the informal notion of effective calculability; the failure of the diagonal argument to lead to an effective but not general recursive function; quasi-empirical evidence. Most importantly, Gandy examines Church's so called step-by-step argument provided for his Thesis (Church 1936, pp. 356-358). That is, when calculating a function in a logic, Church requires each step to be recursive which guarantees the function to be recursive. Where calculating the number theoretic function F in a logic L means the following. L contains expressions $1, 2, 3, ...$ for positive integers, the identity symbol $=$, and a symbol $\{\ \}(\ )$ for the application of a function of one integer to its argument and L has to satisfy certain conditions which amounts to requiring its theorem predicate to be recursively enumerable. Then, according to Church, F is "*calculable within* the logic if there exists an expression f in the logic such that $\{f\}(\mu) = \nu$ is a theorem when and only when $F(m) = n$ is true, μ and ν being the expressions which stand for the positive integers m and n." (p. 357)

Gandy raises concerns about all the kinds of evidence provided, arguing that they are good heuristic arguments but not proofs. About Church's step-by-step argument, for example, he raises the following concern:

> A similar doubt arises in connection with the step-by-step argument. An entirely new kind of algorithm, or an entirely new kind of rule of proof, might proceed by (irreducible) steps which are not recursive. How can one make unassailable predictions about the future development of mathematics? (Turing showed how.) (Gandy 1988, p. 79)

It is clear from the parenthetical remark of "Turing showed how" that Gandy takes Turing's analysis to provide "unassailable" reasons for the claim that the steps in calculating a function must be recursive. And, as a consequence, to constitute a proof of the analogue of Church's Thesis.

Indeed, Gandy, after summarizing Turing's arguments for the boundedness and locality restrictions on human computers, concludes:

> [T]he undeniable limitations [...] show that the change in the record [of the calculation] which is made in a single step can be specified by a *totally finite* (and hence certainly recursive) function. So, without further ado, one can use Church's step-by-step argument. (p. 83)

Thus, Turing's analysis combined with Church's step-by-step argument provides an unassailable proof. According to Gandy, "Turing's work is a paradigm of philosophical analysis: it shows that what appears to be a vague intuitive notion has in fact a unique meaning which can be stated with complete precision." And adds that "[a]fter Turing one can no longer, on this topic, use the ploy of 'it all depends on what you mean'." (p. 86)

It is not entirely clear what Gandy means by proof here. Is Turing's fundamentally philosophical contribution in §9 part of a completely precise mathematical proof? This is definitely not the standard understanding of what a mathematical proof is taken to be. As a consequence, Gandy's understanding of Turing's analysis as a proof of Church's Thesis is not supported by anyone in the literature.

However, it is true that after Turing's analysis, a precise mathematical theorem can be proven. But it has to be emphasized, that such a mathematical theorem is not Church's or Turing's Thesis. Thus, a statement similar to Gandy's "10.3.1. Theorem" can be proved in the mathematical sense, it just will not be Church's Thesis.

To see how such a philosophical analysis can lead to a mathematically precise theorem we take a look at Sieg's (1994 and 2002) where he carefully reconstructs Turing's analysis in §9 of his (1936) in the following way. The analysis makes extensive use of the cognitive limitations of humans while computing in a mechanical fashion. This leads to the boundedness and locality conditions imposed on the computors:[28]

(B) (Boundedness) *There is a fixed bound on the number of configurations a computor can immediately recognize.*

(L) (Locality) *A computor can change only immediately recognizable (sub-)configurations.* (Sieg 2002, p. 396)

And a determinacy condition is also imposed as a consequence of the mechanical nature of the computations considered:

(D) (Determinacy) *The immediately recognizable (sub-)configuration determines uniquely the next computation step.* (p. 397)

Furthermore, the symbolic configurations with which the computers deal are specified and restricted as well. Famously Turing dispenses with the two-dimensional character of regular computations as not essential and restricts the attention to one-dimensional strings; and argues for a finite alphabet.

Once all these conditions and restrictions are precisely described, one can prove what Sieg calls in his (1994):

[28]The word 'computor' is introduced by Gandy for the human computer in Turing's analysis to emphasize that it is a human and not a machine.

> *Turing's Theorem*: Any number-theoretic function F that can be computed by a computor, satisfying the determinacy condition (D) and the [boundedness (B) and locality (L) conditions], can be computed by a Turing machine. (p. 94)

It has to be emphasized again that this mathematically precise and provable theorem is not Church's Thesis. Which also shows that Gandy's rephrasing of Church's Thesis as "10.3.1. Theorem" is far from innocent. Furthermore, while in 'Turing's Theorem' by Sieg all notions (number-theoretic function, Turing machine) and conditions (D, B and L) are precisely defined, in Gandy's "10.3.1. Theorem" there are no conditions imposed on the calculating human being and the notion of 'fixed routine' is not analyzed either.

Gandy claimed that Church's Thesis is already proved by a mathematical proof by Turing. He did not claim that the Thesis is proven by a formal proof, but he did not argue that it is unformalizable either. Now we turn to the formal axiomatic approaches towards Church's Thesis, and to Dershowitz and Gurevich's attempt to formally prove it.

3.3 Formal Axiomatic Approaches to Church's Thesis

In the previous subsection we saw that some people believe Church's Thesis to be provable by a mathematical but not a formal proof. There are people who believe that Church's Thesis is not only provable, but that a formal proof within an axiomatic system can be given. We saw that Kreisel was one of those people who asked for an axiomatic treatment of it and hoped that a formal proof or disproof could be found. He was definitely not alone, and not even the first one to suggest an axiomatic treatment of effective calculability. This line of thought often points to Gödel's views as its origin or, more precisely, to Church's summary of Gödel's attitude from in a letter to Kleene. Before quoting Church's summary of Gödel's view it has to be emphasized that this is not the only possible interpretation of his views, as it will be explained below.

Let us now turn to Gödel's view as reported in a letter of Church's to Kleene, dated November 29, 1935:

> In regard to Gödel and the notions of recursiveness and effective calculability, the history is the following. In discussion [sic] with him the notion of lambda-definability, it developed that there was no good definition of effective calculability. My proposal that lambda-definability be taken as a definition of it he regarded as thoroughly unsatisfactory. [...] His only idea at the time [1934] was that it might be possible, in terms of effective calculability as an undefined notion, to state a set of axioms which would embody the generally accepted properties of this notion, and to do something on that basis. (quoted in Davis 1982, p. 9)

As an example of a more contemporary interpretation coherent with Gödel's phrasing, claiming that Church's Thesis is provable, Shoenfield writes:

> Since the notion of a computable function has not been defined precisely, it may seem that it is impossible to give a proof of Church's Thesis. However, this is not necessarily the case. We understand the notion of a computable function well enough to make some statements about it. In other words, we can write down some axioms about computable functions which most people would agree are evidently true. It might be possible to prove Church's Thesis from such axioms. (Shoenfield 1991, p. 26)

It is this approach Dershowitz and Gurevich claim to follow and champion. That is, they claim in their (2008) that they give a formal axiomatic framework in which Church's Thesis is not only formalizable, but provable as well.

Before turning to Dershowitz and Gurevich's paper, let us take a quick look at the other interpretation coherent with Gödel's view. It is actually not stated by Gödel that the axiomatic treatment has to entail that Church's Thesis is expressible as a provable formula of the axiomatic framework. Indeed, as we saw in Sieg's axiomatic approach, it is possible and worthwhile to give an axiomatic framework for computability based on a precise conceptual analysis without making the Thesis expressible in the framework.[29]

To distinguish between these two axiomatic approaches I adopted Sieg's terminology. The approach in which the Thesis should be a provable formula of a formal axiomatic system given in detail, is called *formal* axiomatics. Thus, Dershowitz and Gurevich's attempt falls under this approach, according to them. The other approach is called *structural* axiomatics, and it is summarized in the case of computability by Sieg in the following way:

> [T]he function of the axiom systems for computing devices can be seen as being similar to that of the axiom systems for the classical algebraic structures like groups, rings or fields, namely, to abstract the essential aspects from a wide variety of instances and point to deep structural analogies. (2008, p. 150)

Sieg introduced Computable Dynamical Systems as the axiomatical characterization for the concept of "mechanical process." Its adequacy (and scope) is shown by a representation theorem, i.e. proving that every function that is computable according to the axioms is also computable by Turing machines. We turn to Dershowitz and Gurevich's work.

Dershowitz and Gurevich's "Formal" Proof of Church's Thesis

In their 2008 paper, *A Natural Axiomatization of Computability and Proof of Church's Thesis*, Dershowitz and Gurevich claim to provide an axiomatic framework for computability and to achieve the following:

[29] Uspensky seems to have a similar approach towards axiomatizing computability in his (1987). However, his conceptual analysis is far from exhaustive and his axioms stay on the informal level.

> [W]e turn Church's Thesis into a precise mathematical statement and explain why the fact that only the recursive functions can be calculated by effective means follows provably from our four postulates. (p. 307)

Thus, with the distinction above, they promise to provide a formal axiomatic framework for computability, in which Church's Thesis is a provable formula.

In the first section of the paper they provide the informal "sketch of axioms" as follows:

> I. *An algorithm determines a sequence of "computational" states for each valid input.*
>
> II. *The states of a computational sequence are structures. And everything is invariant under isomorphism.*
>
> III. *The transitions from state to state in computational sequences are governable by some fixed, finite description.*
>
> IV. *Only undeniably computable operations are available in initial states.*
>
> (p. 306)

They promise to provide formalized versions of these "axioms" in the second and fourth chapter.

They introduce the notion of *state-transition system,* "comprising a set of *states* S, a subset I of which are *initial*, and a (typically partial) *transition function* $\rightsquigarrow$ on states, which determines the next state relation. States with no "next" state, namely, $\{\beta \in S \mid \neg\exists\gamma.\beta \rightsquigarrow \gamma\}$, will be, for us, *terminal states*." (p. 313) With this notion at hand the "formal" version of the first postulate is formulated as follows:

> Postulate I (Sequential time). *An algorithm is a state-transition system. Its transitions are partial functions.* (p. 313)

The "formal" versions of the remaining postulates are quite similar, Postulate IV for example only lists a set of functions that are "undeniably computable" without further arguments. As the mathematical details will not be necessary for our purpose here, I do not present them (though see Postulate IV in the *Remark* below).[30]

State-transition systems that satisfy all four postulates are called *arithmetical algorithm*.[31] In the paper they also introduce the so-called *Abstract State*

[30] Postulates I-III together are called the Sequential Postulates. However, it has to be emphasized that a program of an Abstract State Machine (see the next paragraph) might has several commands that are applicable at the same time. When this occurs, the commands are executed in parallel in one step. ASM-s are thus capable of capturing parallel algorithms as well, for details see Gurevich's (2003).

[31] Although it is not clear what it means for the first postulate, i.e. "*An algorithm is a state-transition system. Its transitions are partial functions.*" to be satisfied by a state-transition system.

Machines or *ASM*-s. When ASM-s satisfy all four postulates they are called *arithmetical ASM*-s. At the end it leads to a somewhat confusing naming convention, because arithmetical algorithms and state-transition systems satisfying all four postulates are used interchangeably with each other.

With these notions in hand the main theorem of the paper, that is, "Church's Thesis", takes the following form:

> THEOREM 4.8 (Church's Thesis). *Every numeric (partial) function computed by an arithmetical algorithm is (partial) recursive.* (p. 327)

However, this statement is clearly not Church's Thesis. This statement presupposes another thesis: namely, that the informal notion of effective calculability is captured by the notion of arithmetical algorithm. This latter thesis, equivalent to that of Church's, is of course not proven. Dershowitz and Gurevich provide as arguments for their thesis a survey of the literature arguing for Church's Thesis. This survey might make for a good motivation or justification of their formal notion, but certainly does not constitute a proof.

Sieg in his (2012) critical review of Dershowitz and Gurevich's paper took issue with their proof of theorem 4.8. This main theorem is a corollary and rephrasal of a preceding theorem, which states that:

> THEOREM 4.5. *A numeric function is partial recursive if and only if it is computable by an arithmetical ASM.* (p. 326)

The proof of this theorem ends with a sketch or hint of an argument:

> [I]t is clear that any arithmetical ASM can be programmed in a standard programming language [...]. Such programs, of course, can compute only partial recursive functions. (p. 326)

A footnote is attached to their remark, arguing that their hint can easily be extended to a complete proof:

> This implicit appeal to the formal effectiveness of standard programming techniques is sometimes also referred to as an invocation of Church's Thesis, but the omitted details could be fleshed out in what amounts to no more than a programming assignment for an undergraduate course. (p. 326, footnote 29)

However, Sieg in his review emphasizes that the allusion to programming techniques is not sufficient to finish the proof:

> For a specific ASM it might be a good programming assignment to show that it calculates a partial recursive function. However, the general claim concerning *any* arithmetical ASM cannot be settled in this way. Rather, the implicit appeal to Church's Thesis is a genuine appeal to a version of Church's Thesis concerning *programming formalisms* for ASMs – that are not given a precise general formulation. This argument has exactly the same semi-circular character as that found in Church's (1936). (p. 119)

Contrary to what they promised, Dershowitz and Gurevich do not provide a formal axiomatization. There is no language provided with inference rules, and Church's Thesis is not presented as a formula. Effective calculability is nowhere introduced, not even as a basic or "undefined" notion. Their approach is closer to a structural approach, characterizing computability by providing a model of computation. Indeed, for example, when they motivate their third postulate, they use a quote of Kolmogorov and Uspensky to describe their approach:

> It seems plausible to us that an arbitrary algorithmic process satisfies our definition of algorithms. We would like to emphasize that we are talking not about a reduction of an arbitrary algorithm to an algorithm in the sense of our definition, but that every algorithm essentially satisfies the proposed definition. (p. 319)

That is a clearly structural characterization of the notion of algorithm.

There are further philosophical and mathematical details that are questionable in Dershowitz and Gurevich's paper. For them, I guide the reader to Sieg's (2012). My one comment on their comparison of their work with that of Gandy and Sieg is the following.

Besides presenting their own development, Dershowitz and Gurevich compare Gandy's and Sieg's computational models (see above) with theirs, and conclude that their presentation is more "generic." As (Sieg 2012) responds to the criticisms extensively, I would like to make one comment on the "genericity" of these models which is not mentioned in Sieg's paper. It is connected to the fourth postulate, whose role, as described in the informal presentation, is to make sure that "only undeniably computable operations are available in initial states." The formal version of the postulate involves the notion of arithmetical state, which is of interest here:

> Definition 4.2 (Arithmetical state) Up to isomorphism, an *arithmetical state* is as follows: Its domain includes the natural numbers **N**, as well as the two (distinct) Boolean truth values, *True* and *False*, and some (other) distinguished value $\perp$ signifying "undefined." Its operations include some or all of the "grade school" operations of arithmetic, namely, zero, successor, addition, subtraction, multiplication, integer quotient, equality, and inequality, as well as logical constants and standard operations for the Booleans. Besides symbols for all these operations, the vocabulary of an arithmetical state may also have various symbols for dynamic functions. (p. 325)

Dershowitz and Gurevich give no explanation why this specific and rather *ad hoc* set of operations is the one allowed and admit that "the choice of the basic functions is somewhat flexible". (p. 324) Furthermore, they did not even indicate what those dynamical functions that might be included could be. What makes those "undeniably computable" and how would we know if we included something non-computable by accident?

On the other hand, as Gandy remarks in his (1980) about his model of computation:

> It is perhaps worth emphasizing how unrestrictive the principles are. Unlike most automata and algorithms which have been proposed, our treatment does not depend on singling out any set of "elementary" operations. (p. 145)

Thus, the model does not have "undeniably computable operations" built in from the beginning. Instead it is the physical limitations and the requirement of proceeding deterministically on discrete finite states that lead to the characterization of computable operations. Sieg's Computable Dynamical Systems preserve this kind of abstractness of the computational model. Hence, Dershowitz and Gurevich's presentation is not more "generic" on the level of the description of the basic operations than Gandy and Sieg, as they start out with a more or less *ad hoc* set of 'certainly' computable functions.

4 Models of Parallel Computing as Computable Dynamical Systems

We saw that Gandy's aim was to capture machine computation including parallel computing as well. In his (1980) paper he showed that Conway's Game of Life can be simulated by his machines. He thought it to be important, as he viewed the Game of Life as an exemplar of a system that Turing's analysis of computability does not cover, but should be shown to be mechanically computable. What distinguishes Conway's Game of Life as a special paradigm of parallel computing from actual physically realizable parallel computers is that it can grow beyond any limit. As summarized at the end of the previous chapter, Sieg in his (2009) showed how to describe the Game of Life as a Computable Dynamical System. However, more realistic cases of parallel computation have not been described in terms of his framework. The goal of this chapter is to supply such a description.

The demonstration that parallel computing can indeed be described in the Computational Dynamical Systems framework is relevant both for machine and human computations. It is clearly relevant for machine computation. As it was already pointed out, there are features of parallel computing that are not directly analyzable in Turing's framework, though we know that what is computable in parallel is also computable by Turing Machines. Furthermore, a non-negligible portion of actual physical computers operate in parallel. Thus, their description in terms of the Computational Dynamical Systems framework is clearly relevant for machine computing. At the same time parallel computing is relevant for human computations as well. The cognitive science community embraced the idea of parallel computing since the 1980s as they saw it as mimicking the "hardware" of the brain. That is, parallel computing is seen as not only trying to reproduce input/output behavior of the brain in terms of computable functions or "software", but also modeling its highly parallel "architecture". This feature of parallel computing will not be further analyzed here, as artificial neural networks were already discussed briefly in the previous section, and the next section is entirely devoted to cognitive models.

The demonstration that parallel computers can be described in the Computable Dynamical Systems framework will take the following form. First a short survey of prevalent models of parallel computation will be given in the next subsection, as there is no single, generally accepted formal model of parallel computing. Then a detailed example will be constructed with the aim to embody several distinctive features of the models of parallel computing. It is described in a similar way to that of Gandy's, De Pisapia's, and Sieg's demonstrations of the Game of Life, Artificial Neural Networks, and Turing Machines being represented in the framework.

However, the demonstration of the features of parallel computing in the Computable Dynamical Systems framework does not constitute a mathematical proof of a representation theorem, such as the demonstration of the computational equivalence of Computable Dynamical Systems to Turing Machines. For,

at the current time the models of parallel computing are not yet fully specified formally, e.g. the set of their admissible operations. This means that the demonstration below, that makes it plausible that all models of parallel computation can be represented in the framework of Sieg's, is the best that can be given as of now. It also points towards possible further work, that is, supplying such a formal description based on the conceptual analysis realistic parallel computing devices. We now turn to the summary of models of parallel computing.

4.1 Models of Parallel Computing

Network Models

These models[32] were used mostly in the early years of the theory of parallel computing. The architecture topology is displayed as a graph. The vertices represent processors and/or memory units while edges display the connections between them. Processors and memory units can communicate directly if they are connected by an edge, otherwise messages have to be sent through intermediate nodes through paths in the graph. Standard topologies among these models are (a) meshes, (b) hypercubes, (c) butterflies, and (d) trees. The following figure displays (a)-(c):

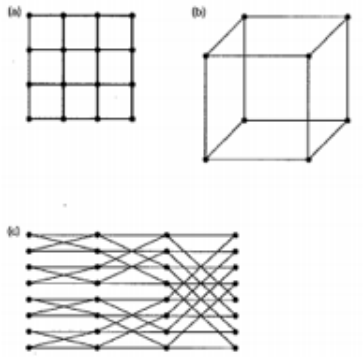


Figure 2, from (Leopold, 2000, p. 202)

Network models have several variants, beyond displaying the different interconnection topologies, depending mostly on what is considered as a step in the computation. When algorithms are analyzed, two parameters are taken into account, the number of processors, P, and the size of the input, N.

These models are used to less extent today, among other reasons, because the programs written for these models lack portability. That is, as the particular architecture is factored into the models, a program written for a particular architecture needs to be reworked if it is being run on another computer. Nevertheless these models can be used to exemplify simpler algorithms and in networks which have fixed special purposes.

[32]This section is based on section 11.1 of Leopold's (2000).

Parallel Random Access Machines, PRAM

The PRAM model was introduced by Fortune and Wyllie in their (1978) and it is a very popular model of parallel computing. It is a shared memory model. The identical processors have access to a global shared memory (and some small local memory for local operations) through a memory access unit (MAU), as shown in the next figure:

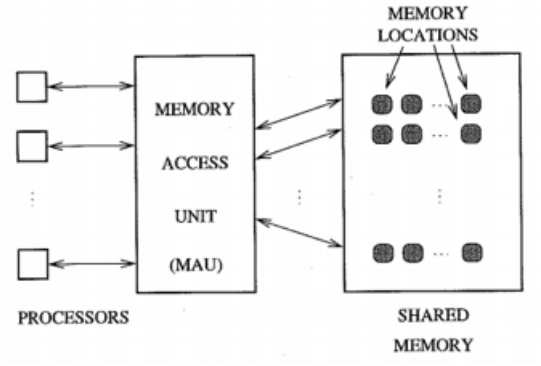


Figure 3, from (Akl, 2000)

A step in the computation of a PRAM algorithm consists of the following operations for each processor:

- reading a datum from shared memory to local memory
- local computations and/or
- writing a datum from local memory to shared

Based on (Leopold, 2000, p. 206), PRAM models are rather popular due to their simplicity and lack of architectural details which leads to comparatively easy algorithm design and analysis. However, the model has been criticized for its lack of reflectivity. There are unrealistic assumptions about the cost of communication and synchronization is enforced after each step without its high cost being taken into account. Moreover, the simulation of one PRAM step on a mesh network can take different O running time than on a hypercubic network. This means that on real architectures algorithms with better O complexity might not outperform algorithms that are supposed to be worse.

Bulk-Synchronous Parallel Model of Computing, BSP

The BSP model was proposed by Valiant (1990) as a bridging model between hardware (architecture) and software (programming) concerns that incorporates more realistic costs of computing and communication. The following quote from Valiant captures the state of practical and theoretical parallel computing at the end of 1980s very well, and also describes Valiant's aim behind the BSP model:

> We take the view that the enabling ingredient in sequential computation is a central unifying model, namely the von Neumann computer. Even with rapidly changing technology and architectural ideas, hardware designers can still share the common goal of realizing efficient von Neumann machines, without having to be too concerned about the software that is going to be executed. Similarly, the software industry in all its diversity can aim to write programs that can be executed efficiently on this model, without explicit consideration of the hardware. Thus, the von Neumann model is the connecting bridge that enables programs from the diverse and chaotic world of software to run efficiently on machines from the diverse and chaotic world of hardware.
>
> Our claim is that what is required before general purpose parallel computation can succeed is the adoption of an analogous unifying bridging model for parallel computation. A major purpose of such a model is simply to act as a standard on which people can agree. In order to succeed in this role, however, the model has to satisfy some stringent quantitative requirements, exactly as does the von Neumann model. Despite the clear benefits that might flow from the adoption of a bridging model, relatively little effort appears to have been invested in discovering one. (Valiant, 1990, p. 104)

So let us take a closer look at Valiant's BSP model. The "bulk-synchronous parallel computer is defined as the combination of three attributes:

- A number of *components*, each performing processing and/or memory functions,
- A *router* that delivers messages point to point between pairs of components, and
- Facilities for synchronizing all or a subset of the components [...] A computation consists of a sequence of *supersteps*. In each superstep, each component is allocated a task consisting of some combination of local computation steps, message transmissions and (implicitly) message arrivals from other components." (1990, p. 105)

Where the "basic task of the router is to realize arbitrary h-relations, or, in other words, supersteps in which each component sends and is sent at most h messages." (p. 105) A schematic picture of a superstep is shown in the next figure:

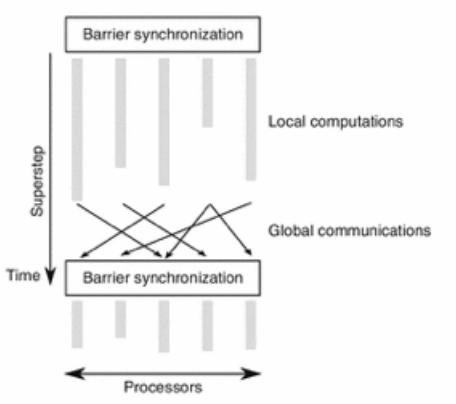


Figure 4, from (Kielmann and Gorlatch, 2011)

Depending on the variants of the BSP models that are being used, the local computations may or may not overlap with global communications. The determination of barrier synchronization also depends on the variant. In Valiant's original paper the facilities for synchronizing check periodically, after L steps, whether a barrier synchronization is in order, but it is not the only way to deal with this issue.

The parameters of the model are the following. It has P physical processors, but programs, in many cases are written for V virtual processors where $P \leq V$. For analyzing performance, h-relations are charged for $g'h + s$ time units, where g' "defines the basic throughput of the router when in continuous use and s the latency startup cost." (p.105) It is assumed that $g'h$ is comparable to s, and thus a g can be chosen where communication costs can be estimated by gh.

The BSP model, intentionally, leaves many design choices open. For example, both single and multiple instruction streams are allowed, that is, the different processors do not necessarily execute the same instructions. It also allows, in the multiple instruction stream case, for some processors to not take part in particular barrier synchronizations. Furthermore, as "the BSP model separates computation from communication, no particular network topology is favored beyond the requirement that a high throughput be delivered." (p. 109) The BSP model can be considered as a generalization of the PRAM model, that is more realistic and has several advantages (Skillicorn et al., 1996). And "[c]ompared with network models, BSP simplifies algorithm design and enables portability. [...] BSP accounts for the cost of communication and synchronization. The model is reasonably reflective of a wide class of architectures." (Leopold, 2001, p. 211)

The BSP model has several improved, more realistic versions. For example, Juurlink and Wijshoff (1996) introduced the Extended BSP model, E-BSP, where the communication is not as balanced as it is assumed by the standard BSP model. It also takes some details of the architecture into account. The BSP model served also as a starting point for the introduction of the later LogP model in (Culler at al., 1996).

4.2 Description of Parallel Models as Computable Dynamical Systems

The aim of the following, rather long and detailed example is to indicate how the features of parallel computing summarized in the subsections above can be accommodated in Sieg's Computable Dynamical Systems framework. That is, in this example I will show how to accommodate global and local communication, memory access units (MAU), synchronization, and make remarks on how limited communication and running different programs on different processors could be described in this framework. The example serves only this purpose, it is not a useful or realistic computational problem, nor is its solution an optimal one. Its sole purpose is to make (if only limited) use of every item from the above list. Furthermore, although the example is rather detailed, some rules are only mentioned or indicated, as giving a complete description would take even longer without further merit. Now let us turn to the example:

> The problem to be solved is the following: the input is a decreasing sequence of at most t integers where none of the integers are bigger than k. The output is obtained by adding 1 to each integer and then summing them.

Parallelism will be utilized during adding 1 to each integer in the input by separate processors. The device is designed to have insufficient memory, which in some cases leads to the need of internal, local and global, communication between its processors and synchronization between them.

The device, as shown in Figure 5, is described as follows: it consists of a *main processor* and several *local processors*. The main processor is the one that receives the input, outputs the solution, and directs the local processors via global communication. The main processor has a large *main memory* and t *small memory units*. It stores the input in its main memory in "unary code," i.e. the integer n is represented by n 1-s and there are 0-s between the integers. The end of the input is denoted by two consecutive 0-s.

In the first step the main processor copies the input from the main memory to the small memory units. Each small memory unit stores exactly one integer from the input. This process is sequential as it happens within one processor. Then the input is copied in parallel from the small memory units to the memory of the local processors. Thus small memory units behave as a MAU. Both the small memory units and the memories of the local processors can store only k bits of information. This leads to a problem when the local processors add 1 (in parallel) to their input, if they received k as their input. In that case adding 1 would lead to *overflow* which will be indicated by storing a + sign instead of a 1 in the last memory square. As the time needed for the process of adding 1 depends on the length on the input, the local processors have to be *synchronized* to begin addressing problems arising from overflows simultaneously.

Synchronization is directed by the *clock* in the main processor. Once synchronized, the overflow problem will be dealt with by local communication between the local processors. If there is not enough space to handle the overflow

then the local processor with index 1 will send out a global *error* message. If the overflow is resolved, then the integers are copied back to the main processor, first to the small memory units in parallel, and, finally to the main memory.

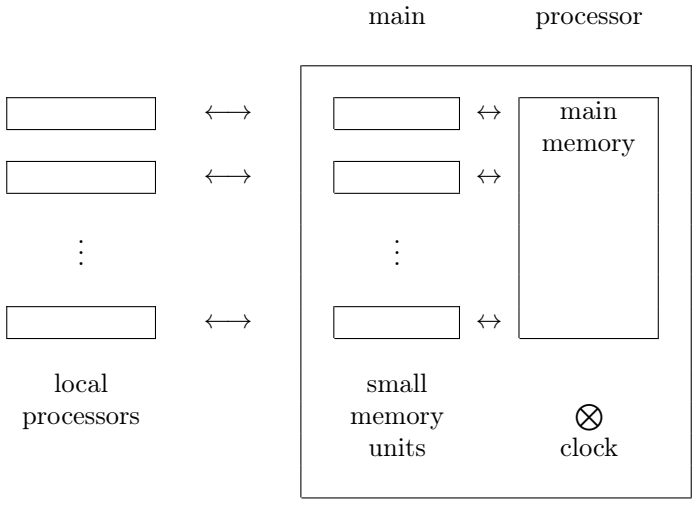


Figure 5

Before turning to the description of the device, some conventions are needed in the notation to make it easier to parse:

> Conventions: (1) lower case letters are atoms; (2) lower case letters with a bar, i.e. $\overline{x}$, are of the form r^k where r is an atom and k is a fixed integer; (3) the numbers 0 and 1 and the symbols $*$ and $+$ are also given in the form r^k where r is an atom and k is a fixed integer.

We will call the following structure the 'chain' of the *natural numbers until* k:

$$\mathbb{N}_k = \{\langle a_1, a_1, \overline{n}\rangle, \langle a_1, a_2, \overline{n}\rangle, \langle a_2, a_3, \overline{n}\rangle, \ldots, \langle a_{k-2}, a_{k-1}, \overline{n}\rangle, \langle a_{k-1}, a_k, \overline{n}\rangle\}^{33}$$

where we identify a_i with the number i and denote it by i' ($a_i \sim i'$) to show that it is encoded by an atom. Thus 1 and $1'$ have different structures set-theoretically to distinguish 1, the symbol that the computer reads from $1'$, the member of the sequence of natural numbers represented by atoms within the framework.

This chain structure is rather weak, it only captures the successor/predecessor structure. That is, once $5'$ is singled out, the chain tells us that $4'$ is the preceding and $6'$ is the succeeding element, but we have no further structural properties, e.g. divisibility etc. However, we can single out any number, like $5'$ by "hardcoding" it, that is, including the whole initial segment of the chain until $5'$. This structure is very useful when we would like to use numbers and

[33] If somehow the last one needs to be singled out in a special way, we can use $\langle a_k, a_k, a_k, \overline{n}\rangle$ but not $\langle a_k, a_k, \overline{n}\rangle$ because it cannot be distinguished from $\langle a_1, a_1, \overline{n}\rangle$.

indices in a rule in general, or the relationship between them, e.g. that $i = j+2$. Encoding the chain structure of natural numbers by atoms allows us in general to use one rule when dealing with numbers[34] instead of multiplying (essentially) the same rule for all numbers.

Now we turn to the description of the computing device in the framework of discrete dynamical systems:

Description of the parallel computer

-Description of the *main processor*:

Overall structure:

$$\{\langle g, \overline{g}\rangle, \langle g, \overline{q_i}\rangle, \langle g, m_0\rangle, \langle g, m_1\rangle, \ldots, \langle g, m_t\rangle\}$$

where g is the atom for the main or 'global' processor and $\overline{g}$ is used to encode this function, and $\overline{q_i}$ is its current state. m_0 is the atom of the *main memory* within the main processor while $m_1, \ldots, m_t$ are the atoms of the *MAU memory units*.

The main memory is the union of the following sets:

$\{\langle m_0, \overline{m_m}\rangle\}$,
$\{\langle m_0, k_1, k_1\rangle, \langle m_0, k_1, k_2\rangle, \ldots, \langle m_0, k_s\rangle, \langle k_j, *\rangle\}$, i.e. the 'tape'

where m_0 is the atom for the main memory, k_i are the memory units or "squares on the tape" and the $*$ denotes where the reading 'head' is currently. The memory is ordered sequentially and its content is given by:

$$\{\langle k_1, 1\rangle, \langle k_2, 1\rangle, \langle k_3, 0\rangle, \ldots\}$$

where, again, 0 and 1 are not atoms.

The small memory units (MAU) are described as follows:

$$\{\langle m_j, j', \overline{m_s}\rangle, \langle m_j, l_i, 1'\rangle, \langle m_j, l_{i+1}, 2'\rangle, \ldots, \langle m_j, l_{i+k-1}, k'\rangle\}$$

where m_j is the atom for the jth small memory unit, its index is indicated by j' and $\overline{m_s}$ shows that it is a small memory unit. The further triples indicate the sequential structure or indexing of the "squares" in the memory, currently without content.[35]

Description of the *clock*:

$$\{\langle c, \overline{c}, n'\rangle, \langle g, c\rangle, \langle c, \overline{q_t}\rangle\}$$

where c is the atom for the clock, $\overline{c}$ encodes it to be a clock, and n' is the "time" the clock shows. It will have two internal states, $\overline{q_t}$ for 'time mode' and $\overline{q_s}$ for 'synchronization mode.' These inner states will always appear in an ordered pair with the atom c. The pair $\langle g, c\rangle$ is included in the description to indicate that the clock is part of the main processor.

[34] With a possible addition of rules for the first and last number or index.

[35] It is important to emphasize that the indices i and j are independent. The indices $i, i+1, ..., i+k-1$ are there only to help the reader in parsing the rules below. The could have been just denoted by different atoms $a, b, ..., o$ as the sequential structure is indicated by $1', 2', ..., k'$ already.

-Description of the *local processors*:

$$\{\langle p_i, i', \overline{p_l}\rangle, \langle p_i, \overline{o_f}\rangle \langle p_i, \overline{q_k}\rangle, \langle p_i, o_s, 1'\rangle, \langle p_i, o_{s+1}, 2'\rangle, \ldots, \langle p_i, o_{s+k-1}, k'\rangle\}$$

where p_i is the atom for the ith local processor, its index is indicated by i' and $\overline{p_l}$ shows that it is a local processor.[36] As the processors are autonomous computing devices on their own, they have their own inner states, that is, $\overline{q_k}$. However, in the beginning they are "turned off," that is why they have another binary internal state $\overline{o_f}$. The local processors will start computing once they are "turned on," that is, when their internal state is set to $\overline{o_n}$. The further triples indicate the sequential structure or indexing of the squares in the memory, currently without content.

Finally a remark must be added considering the description of the device. This description is clearly *redundant*. For example, in $\langle m_j, l_{i+1}, 2'\rangle$ the $2'$ indicates that l_{i+1} is the second memory square in the j-th small memory unit. It is clear that the pair $\langle m_j, 2'\rangle$ would suffice for the same purpose. However it would make the description harder to parse. It would also hide the fact that there is a concrete memory location which has an index or address, that is, it would be less faithful to a concrete computing device. As Gandy's aim was to account for the device as well, I decided to use this description.

Now we can turn to the description of the rules of the algorithm:

Rules of the algorithm

As the rules for such parallel device are rather complex and long, conventions for the presentation are needed. All causal neighborhoods and determined regions are sets of ordered tuples, for transparency the opening and closing brackets will appear one line before and after the elements of the set.[37]

As the content of the sets needs explanation, borrowing from notation in programming languages, the role of certain elements of the set will be explained after a # sign. The comments after the sign in ordinary language are not part of the set, naturally.

As in many cases a large part of the causal neighborhood serves the purpose of singling out particular pieces from the device, a large part of those have to be included in the determined region as well, they have to be "put back" into the device, since those are fixed parts of it. To make determined regions easier to parse, their parts which are shared with the causal neighborhoods will appear at the end of these rules, and, with a notation borrowed from music scores, between $\| :$ and $: \|$ symbols. As mentioned above, these are usually describing parts and structure of the device itself.

Let us now turn to the rules of the algorithm itself:

Moving the input within the main processor. In the beginning we set the inner state of the main processor to q_1, that is, we add the ordered pair

[36] Here, again, the indices i and s are independent. See the previous footnote.

[37] In most cases. Exceptions are made in cases of very short and simple rules.

$\langle g, \overline{q_1} \rangle$ to the description. We can "understand" q_1 as meaning 'copying/moving input from the main memory to the small memory units.' As the copying happens within the main processor, i.e. within one processor, it is going to be a sequential process.

We also add the pairs: $\langle k_1, * \rangle$ and $\langle l_j, * \rangle$. The $*$-s indicate where the scanning/reading 'head'-s are. At the beginning k_1 is the first memory square of the main memory, while l_i is the first memory square of the small memory unit with the highest index. At the start, we can pick out the atoms k_1 and l_i "by hand" without any problems.[38]

Start:
Cn:

{

$\langle k_1, * \rangle, \langle l_j, * \rangle, \langle k_1, 1 \rangle,$ #the 'head'-s and the input
$\langle m_0, k_1, k_1 \rangle, \langle m_0, \overline{m_m} \rangle, \langle g, \overline{q_1} \rangle,$ #the relevant part of the device
$\langle m_t, l_j, 1' \rangle,$ #where to move the input
$\langle 1', 2', \overline{n} \rangle, \langle m_t, l_{j+1}, 2' \rangle, \langle m_0, k_1, k_2 \rangle$ #where the 'head'-s will be moved

}

Dr:

{

$\langle l_i, 1 \rangle, \langle k_2, * \rangle, \langle l_{i+1}, * \rangle,$ #copying the input and moving the 'head'-s
$\| : \langle g, \overline{q_1} \rangle, \langle m_0, \overline{m_m} \rangle, \langle 1', 2', \overline{n} \rangle, \langle m_0, k_1, k_1 \rangle,$ #tuples common with Cn
$\langle m_0, k_1, k_2 \rangle, \langle m_t, l_i, 1' \rangle, \langle m_t, l_{i+1}, 2' \rangle : \|$

}

If the aim would be to keep the input instead of erasing it upon reading, the ordered pair $\langle k_1, 1 \rangle$ should be included in the Dr.[39]

In the steps following the initial move two cases have to be distinguished considering the movement of the 'head' within the small memory units: (1) simply moving it to the right; (2) moving "down" to the first memory square of the unit with the preceding index. Depending on whether a 0 or a 1 is being read from the main memory we get several different cases. Below I only write out some of them, as there are many of them.

(1) If the 'head' in the main memory unit reads a 1 and there is a square with a larger index in the current small memory unit:

[38] But we could actually put it in by a rule as well, since, for example in the case of k_1 there is only one atom which appears in ordered tuples with the following structure $\{\langle m_0, k_1, k_1 \rangle, \langle m_o, \overline{m_m} \rangle\}$.

[39] The use of Dr is not entirely precise here. Instead, it should be called $G(Cn)$, as Dr is the set of all $G(Cn)$-s applied in parallel. I chose to use Dr as it is easier to distinguish from Cn than $G(Cn)$.

Cn:

{

$\langle k_i, * \rangle, \langle l_j, * \rangle, \langle k_i, 1 \rangle,$ #the 'head'-s and the input

$\langle m_0, \overline{m_m} \rangle, \langle g, \overline{q_1} \rangle,$ #the relevant part of the device

$\langle m_t, l_j, n' \rangle,$ #where to move the input

$\langle n', (n+1)', \overline{n} \rangle, \langle m_t, l_{j+1}, (n+1)' \rangle,$ #where the 'head'-s will be moved

$\langle m_0, k_i, k_{i+1} \rangle$ #where the 'head'-s will be moved

}

Dr:

{

$\langle l_{j+1}, 1 \rangle, \langle k_{i+1}, * \rangle, \langle l_{j+1}, * \rangle,$ #copying the input and moving the 'head'-s

$\| : \langle g, \overline{q_1} \rangle, \langle n', (n+1)', \overline{n} \rangle, \langle m_t, l_j, n' \rangle,$

$\langle m_t, l_{j+1}, (n+1)' \rangle, \langle m_0, k_i, k_{i+1} \rangle, \langle m_0, \overline{m_m} \rangle : \|$

}

(2) If the 'head' is on the last square of the small memory unit and the main memory square is on a 0:

Cn:

{

$\langle k_i, * \rangle, \langle l_j, * \rangle, \langle k_i, 0 \rangle,$ #the 'head'-s and the input

$\langle m_0, \overline{m_m} \rangle, \langle g, \overline{q_1} \rangle, \langle m_0, k_i, k_{i+1} \rangle,$

$\langle m_t, l_j, n' \rangle,$ #l_j is the last memory square

$\langle 1', 1', \overline{n} \rangle, ..., \langle (n-1)', n', \overline{n} \rangle,$ #l_j is the last memory square

$\langle m_{t-1}, (t-1)', \overline{m_s} \rangle, \langle m_{t-1}, l_x, 1' \rangle,$ #the first memory square

$\langle m_t, t', \overline{m_s} \rangle, \langle (t-1)', t', \overline{n} \rangle$ #of the memory unit with smaller index

}

Dr:

{

$\langle k_{i+1}, * \rangle, \langle l_x, * \rangle,$ #moving the 'head'-s

$\langle g, \overline{q_2} \rangle,$ #new inner state indicating that a 0 was seen

$\| : \langle m_0, \overline{m_m} \rangle, \langle m_0, k_i, k_{i+1} \rangle, \langle m_t, l_j, n' \rangle,$

$\langle 1', 1', \overline{n} \rangle, ..., \langle (n-1)', n', \overline{n} \rangle,$

$\langle m_{t-1}, (t-1)', \overline{m_s} \rangle, \langle m_{t-1}, l_x, 1' \rangle, \langle m_t, t', \overline{m_s} \rangle, \langle (t-1)', t', \overline{n} \rangle : \|$

}

In general, every time a 0 is seen in the main memory, the inner state of the main processor is changed to q_2. If the next bit in the main memory is a 1, then the inner state changes back to q_1. However, if it is another 0, indicating that we reached the end of the input, the internal state changes to q_3, meaning that "the input is ready to be moved to the local processors."

Moving the input to the local processors. Now we assume that the input is copied from the main memory to the small memory units and that the inner state of the main processor is q_3. In the next step the content of the small memory units are copied into the memory locations of the local processors. This process is done in parallel, i.e. it takes only one step. At the same time the local processors are set to be "on" by changing their inner state to o_n. Furthermore,

'head'-s are placed on every local processors first memory square. To make the rules more transparent, we present these as two separate rules.

Copying the input to the local processors in one step:

Cn:

{
$\langle g, \overline{q_3} \rangle$ #input is ready to be moved to local processors
$\langle m_i, i', \overline{m_s} \rangle, \langle m_i, l_j, k' \rangle,$ #a small memory unit location
$\langle l_j, 1 \rangle,$ #which contains a 1
$\langle p_i.i'.\overline{p_l} \rangle, \langle p_i, o_s, k' \rangle$ #local processors memory location with same indices
}

Dr:

{
$\langle o_s, 1 \rangle,$ #1 moved to the local processors memory
$\| : \langle m_i, i', \overline{m_s} \rangle, \langle m_i, l_j, k' \rangle, \langle p_i.i'.\overline{p_l} \rangle, \langle p_i, o_s, k' \rangle : \|$
}

Although i'-s and k'-s are only atoms, $\overline{m_s}$ and $\overline{p_l}$ picks out the proper place from the device. Since as indices they are identical in the small memory units and the local processors, the $\in$-isomorphism "takes" each 1-s from the small memory units to local processors in the right way in one step.[40]

Now the rule that turns "on" the local processors and places a 'head' on their first memory square is described:

Cn:

{
$\langle g, \overline{q_3} \rangle, \langle p_i, \overline{o_f} \rangle,$ #local processors ready to be turned "on"
$\langle p_i, i', \overline{p_l} \rangle, \langle p_i, o_x, 1' \rangle, \langle 1', 1', \overline{n} \rangle$ #first memory square of a local processor
}

Dr:

{
$\langle p_i, \overline{o_n} \rangle, \langle p_i, \overline{q_4} \rangle,$ #the local processor is turned "on"
$\langle o_x, * \rangle,$ #a 'head' on its first memory square
$\| : \langle p_i, i', \overline{p_l} \rangle, \langle p_i, o_x, 1' \rangle, \langle 1', 1', \overline{n} \rangle : \|$
}

The new inner state, i.e. q_4, is now in one ordered tuple with p_i the local processor. More precisely, with the local processors, as each one of those has such an ordered pair now. It is assigned to the local processors as the main processor does not have an impact on the computations of the local processors, beyond forcing them into synchronization.

[40] One can possibly claim that moving the whole input in one step is irrealistic. To make this process more realistic: a limit could easily be imposed on the size of communication, say h bits per local processor. Before moving anything from the small memory units we could put 'head'-s on the first memory squares of the small memory units and the local processors. Then we would only need two new rules: (1) copy the next h units starting from the head and move the head $h + 1$ units; (2) dealing with the end of the memory units.

Adding plus 1. Now that only the local processors have internal states the main processor is not doing anything beyond running a clock that will force synchronization, see below. The role of the local processors is to add plus 1 to their input integer. That is, the local processors have to reach the end of their input and then put an extra 1 after the last one. However, if their input is k, then it will lead to overflow since they have only k memory squares. Overflow will be indicated by changing the last 1 into a + sign.

In this setting the internal state q_4 can be understood as "saw a 1 and moved to the right." Even when the 'head'-s are on the first square it will not lead to problems. There are two general rules with almost identical causal neighborhoods. Below, both causal neighborhoods will be indicated within one set and there will be two determined regions displayed immediately after:

Cn:

{

$\langle p_i, \overline{o_n} \rangle, \langle p_i, \overline{q_4} \rangle,$ #the local processor is "on"

$\langle p_i, o_x, n' \rangle, \langle o_x, * \rangle,$ #place of the 'head'

$\langle o_x, 1 \rangle \quad OR \quad NOTHING,$ #content of the square

$\langle p_i, o_{x+1}, (n+1)' \rangle, \langle n', (n+1)', \overline{n} \rangle$ #the square to the right

}

If the causal neighborhood contains $\langle o_x, 1 \rangle$, that is, a 1 is being read and there is a square to the right, just move the 'head' to the right and "put back" everything else:

Dr:

{

$\langle o_{x+1}, * \rangle,$ #move the 'head' to the right

$\| : \langle p_i, \overline{o_n} \rangle, \langle p_i, \overline{q_4} \rangle, \langle o_x, 1 \rangle,$

$\langle p_i, o_x, n' \rangle, \langle p_i, o_{x+1}, (n+1)' \rangle, \langle n', (n+1)', \overline{n} \rangle : \|$

}

If the causal neighborhood does not contain $\langle o_x, 1 \rangle$, then write a 1 into the current square, i.e. complete adding plus 1. This also means that the processor has to change internal state, acknowledging that it finished its task:

Dr:

{

$\langle o_x, 1 \rangle,$ #added plus 1

$\langle p_i, \overline{q_5} \rangle,$ #new inner state, "added plus 1"

$\| : \langle p_i, \overline{o_n} \rangle, \langle o_x, * \rangle, \langle p_i, o_x, n' \rangle, \langle p_i, o_{x+1}, (n+1)' \rangle, \langle n', (n+1)', \overline{n} \rangle : \|$

}

For the second determined region in itself there would have been no need to determine whether there is another square to the right. However, in this way on of the two causal neighborhoods above is strictly a subset of another, thus according to our convention, always just one of them applies.

The only remaining case that has to be dealt with is when the 'head' is on the last memory square. There are two sub-cases again, depending on whether

the square is empty or contains a 1. The latter leads to overflow. As the former is unproblematic it will only be indicated in a comment following the rule. Again, recognize that the following causal neighborhood is a strict subset of the previous ones, thus if there is a square to the right, then always the above rule applies and not the following one.

The rule that applies in the case when the 'head' is on the last memory square and it contains a 1:

Cn:

$\{\langle p_i, \overline{o_n}\rangle, \langle p_i, \overline{q_4}\rangle, \langle o_x, *\rangle, \langle o_x, 1\rangle\}$

Dr:

$\{\langle o_x, +\rangle,$ #+ sign indicating overflow
$\langle p_i, \overline{q_5}\rangle,$ #new inner state, "added plus 1"
$\| : \langle p_i, \overline{o_n}\rangle, \langle o_x, *\rangle : \|\}$

In the case when the last square is empty, $\langle o_x, 1\rangle$ does not appear in the causal neighborhood. Instead, it replaces $\langle o_x, +\rangle$ in the determined region.

This completes the "adding plus 1" process of the local processors. We see that once a processor is finished it gets into state q_5, becomes ready to treat possible overflows, but there are no further rules with inner state q_5. Furthermore, it does not "know" whether the other processors are already finished or not, as the time needed for adding plus 1 depends on the size of the input. That is why synchronization is needed.

Synchronization. To accommodate synchronization, a clock was included in the main processor. It will be periodic, that is, it will "count" until some fixed unit of time, then force the local processors to stop, then synchronize them, and, finally, start "counting" from 1 again. There will be two internal states of the clock, q_t for 'time mode' and q_s for 'synchronization mode.'

To achieve this two rules are needed. A rule for the time passing in the clock in general:

Cn:

$\{$
$\langle c, \overline{q_t}\rangle,$ #clock is in 'time mode'
$\langle c, \overline{c}, n'\rangle,$ #the "current time"
$\langle n', (n+1)', \overline{n}\rangle$ #the next number
$\}$

Dr:

$\{$
$\langle c, \overline{c}, (n+1)'\rangle$ #the next second
$\| : \langle c, \overline{q_t}\rangle, \langle n', (n+1)', \overline{n}\rangle : \|$ #stay in 'time mode'
$\}$

And a rule for the last second in the period. However, after the last second the clock will not immediately count again from the first second. The clock will be set to the first second, but before "counting" again it will go into 'synchronization mode' to handle the local processors:

Cn:

{

$\langle c, \overline{q_t} \rangle$, #clock is in 'time mode'

$\langle c, \overline{c}, n' \rangle$, #the "current time"

$\langle 1', 1', \overline{n} \rangle, \langle 1', 2', \overline{n} \rangle, ..., \langle (n-1)', n', \overline{n} \rangle$, #reached the end of the period

$\langle n', (n+1)', \overline{n} \rangle$ #the next number

}

Remark: here there is no need for $\langle n', (n+1)', \overline{n} \rangle$ to be part of the causal neighborhood. However, including it ensures that the causal neighborhood of this rule contains the causal neighborhood of the previous one. Thus, when the clock reaches the end of the period, this is the rule that gets used and not the previous one, because of the convention of always the rule with the largest causal neighborhood gets applied.

Dr:

{

$\langle c, \overline{q_s} \rangle$, #clock is in 'synchronization mode'

$\langle c, \overline{c}, 1' \rangle$, #the time when it is going to be 'time mode' again

$\| : \langle 1', 1', \overline{n} \rangle, \langle 1', 2', \overline{n} \rangle, ..., \langle (n-1)', n', \overline{n} \rangle, \langle n', (n+1)', \overline{n} \rangle : \|$

}

Now that the clock is in 'synchronization mode' the local processors should be forced to stop computing. The rule for turning "off" the local processors for synchronization looks as follows:

Cn:

{

$\langle c, \overline{q_s} \rangle$, #clock is in 'synchronization' mode

$\langle p_1, \overline{o_n} \rangle, \langle p_2, \overline{o_n} \rangle, ..., \langle p_t, \overline{o_n} \rangle$ #listing every local processor

}

Dr:

{

$\langle p_1, \overline{o_f} \rangle, \langle p_2, \overline{o_f} \rangle, ..., \langle p_t, \overline{o_f} \rangle$ #turning 'off' every local processor

$\| : \langle c, \overline{q_s} \rangle : \|$ #staying in 'synchronization' mode

}

There are two cases. The first is when all local processors finished adding plus 1 and thus are in internal state q_5. In this case their internal state should be set to q_6, indicating that they are now ready to deal with overflow:

Cn:

{

$\langle c, \overline{q_s} \rangle$,

$\langle p_1, \overline{o_f} \rangle, \langle p_2, \overline{o_f} \rangle, ..., \langle p_t, \overline{o_f} \rangle$, #every processor is 'off'

$\langle p_1, \overline{q_5} \rangle, \langle p_2, \overline{q_5} \rangle, ..., \langle p_t, \overline{q_5} \rangle$ #every processor finished adding plus 1

}

Dr:

{

$\langle c, \overline{q_t} \rangle$, #back to 'time' mode

$\langle p_1, \overline{o_n} \rangle , \langle p_2, \overline{o_n} \rangle , ..., \langle p_t, \overline{o_n} \rangle$, #every processor is set back to 'on'

$\langle p_1, \overline{q_6} \rangle , \langle p_2, \overline{q_6} \rangle , ..., \langle p_t, \overline{q_6} \rangle$ #every processor is set to deal with overflow

}

The second case is when there is at least one local processor that have not finished yet. In this case all local processors are set back to "on" to continue their process (if they have not finished yet):

Cn:

{

$\langle c, \overline{q_s} \rangle$,

$\langle p_1, \overline{o_f} \rangle , \langle p_2, \overline{o_f} \rangle , ..., \langle p_t, \overline{o_f} \rangle$, #every processor is 'off'

}

Dr:

{

$\langle c, \overline{q_t} \rangle$, #back to 'time' mode

$\langle p_1, \overline{o_n} \rangle , \langle p_2, \overline{o_n} \rangle , ..., \langle p_t, \overline{o_n} \rangle$, #every processor is back to 'on'

}

There are two remarks in order: (1) here, again, we take advantage of the convention that the rule largest causal neighborhood applies. If every local processor finished adding plus 1, then both the previous and the latter causal neighborhoods appear in the device, but the previous one clearly contains the latter one, thus the previous rule has to be applied. (2) as the other "relevant" internal state of the local processors were not cut out, upon turning "on" the processors, each of them will "return" to its internal state before the synchronization, regardless of what that state was (i.e. regardless whether they already finished or not).

* * *

Alternative solution for dealing with the local processors

During synchronization the processors could be set to inner state q_6 1-by-1 if they are finished adding plus 1, and can keep them "off" until every one of them is finished. The rules could be modified to fit this scenario in the following way.

For the finished processors the following rule applies:

Cn:

{$\langle c, \overline{q_s} \rangle$,

$\langle p_i, \overline{o_f} \rangle , \langle p_i, \overline{q_5} \rangle$ #the i-th processor is finished

}

Dr:

{$\langle c, \overline{q_t} \rangle$,

$\langle p_i, \overline{o_f} \rangle , \langle p_i, \overline{q_6} \rangle$ #set to $\overline{q_6}$ but still 'off'

}

For those that are not yet finished:

Cn:
$\{\langle c, \overline{q_s}\rangle, \langle p_i, \overline{o_f}\rangle\}$

Dr:
$\{\langle c, \overline{q_t}\rangle, \langle p_i, \overline{o_n}\rangle\}$ #the i-th processor is turned back 'on'

Remark: here, again, the convention was utilized, the latter rule applies only in case p_i is not in q_5.

In this case different processors would do "different" things at the same time. This shows in general how processors could be taken out of synchronization, can be "reached" individually, and are capable of running different subroutines at the same time. This shows that this framework is capable of describing more realistic parallel computer networks as well.

* * *

Dealing with overflow. The remaining task is to take care of overflows if possible. If it is not possible, then a global error message will be sent out.

The idea of dealing with the overflow is the following. Overflow occurs at the processors with larger indexes. More precisely, if there is overflow, then the local processor with the largest index has to have an overflow, because the input was a decreasing sequence of integers. This means that overflow can be resolved if the processors with lower indices have empty memory squares. Thus the + signs indicating overflow will be "pushed down" to the processors with lower indices. This will be done by local "communication" between the processors. Each processor with overflow will "determine" if the local processor with the previous index does or does not have an overflow. If it does have, then nothing happens in this step. If the other processor does not have an overflow, then the + sign is being moved "down." More precisely: (1) the + sign is removed from the processor on the top, and (2) if the processor in the bottom stores a number smaller than k then 1 is added there, while if it stores k, then the content of its last memory square is changed from 1 to a + sign. This subroutine is finished either when there are no more + signs, or when the processor with index 1 has an overflow. This latter indicates that there is not enough memory space to deal with overflows, hence a global error message will be sent out. If there are no + signs left, the contents of the memory squares of the local processors have to be copied back to the small memory units and then to the main memory.

The important rules in this task are the rules for cases when there is a 1 "under" a + sign. Two cases can be distinguished, depending on whether the 1 is on the last memory square or it appears before it.

In case both the + sign and the 1 are on the last memory square of the local processor, i.e. the + sign is being "pushed down":

Cn:

{

$\langle p_i, i', \overline{p_l}\rangle, \langle p_i, \overline{o_n}\rangle, \langle p_i, \overline{q_6}\rangle,$ #the i-th processor

$\langle p_{i-1}, (i-1)', \overline{p_l}\rangle, \langle p_{i-1}, \overline{o_n}\rangle, \langle p_{i-1}, \overline{q_6}\rangle,$ #the $(i-1)$-th processor

$\langle (i-1)', i', \overline{n}\rangle,$

$\langle p_i, o_x, n'\rangle, \langle o_x, *\rangle, \langle o_x, +\rangle,$ #overflow

$\langle p_{i-1}, o_y, n'\rangle, \langle o_y, *\rangle, \langle o_y, 1\rangle$ #1 in the bottom

}

Dr:

{

$\langle o_x, 1\rangle, \langle o_y, +\rangle,$ #1 on the top, overflow in the bottom

$\| : \langle (i-1)', i', \overline{n}\rangle, \langle p_i, i', \overline{p_l}\rangle, \langle p_i, \overline{o_n}\rangle, \langle p_i, \overline{q_6}\rangle, \langle p_i, o_x, n'\rangle, \langle o_x, *\rangle,$

$\langle p_{i-1}, (i-1)', \overline{p_l}\rangle, \langle p_{i-1}, \overline{o_n}\rangle, \langle p_{i-1}, \overline{q_6}\rangle, \langle p_{i-1}, o_y, n'\rangle, \langle o_y, *\rangle : \|$

}

In case the 1 is not in the last memory square of the local processor, i.e. when this overflow is resolved:

Cn:

{

$\langle p_i, i', \overline{p_l}\rangle, \langle p_i, \overline{o_n}\rangle, \langle p_i, \overline{q_6}\rangle,$ #the i-th processor

$\langle p_{i-1}, (i-1)', \overline{p_l}\rangle, \langle p_{i-1}, \overline{o_n}\rangle, \langle p_{i-1}, \overline{q_6}\rangle,$ #the $(i-1)$-th processor

$\langle p_i, o_x, n_1'\rangle, \langle o_x, *\rangle, \langle o_x, +\rangle,$ #overflow

$\langle p_{i-1}, o_y, n_2'\rangle, \langle o_y, *\rangle, \langle o_y, 1\rangle$ #1 in the bottom

$\langle p_{i-1}, o_{y+1}, (n_2+1)'\rangle,$ #memory location to the right in the bottom

$\langle (i-1)', i', \overline{n}\rangle, \langle n_2', (n_2+1)', \overline{n}\rangle$

}

Dr:

{

$\langle o_{y+1}, *\rangle,$ #moving the bottom 'head' to right

$\langle o_x, 1\rangle, \langle o_{y+1}, 1\rangle,$ #overflow resolved

$\| : \langle p_i, i', \overline{p_l}\rangle, \langle p_i, \overline{o_n}\rangle, \langle p_i, \overline{q_6}\rangle, \langle p_i, o_x, n_1'\rangle, \langle o_x, *\rangle,$

$\langle p_{i-1}, (i-1)', \overline{p_l}\rangle, \langle p_{i-1}, \overline{o_n}\rangle, \langle p_{i-1}, \overline{q_6}\rangle, \langle p_{i-1}, o_y, n_2'\rangle,$

$\langle (i-1)', i', \overline{n}\rangle, \langle n_2', (n_2+1)', \overline{n}\rangle : \|$

}

Remark: in case there is a clash between $n_1, n_2, i, i-1$ a small (finite) number of further rules are needed, because $\in$-isomorphisms are used.

It has to be pointed out that there is no need for rules for the case when there are two + signs above each other. For, in that case everything has to be put back, as it leads to no change. At the same time, the + sign "below" might moves "down" if it has a 1 under it, which would lead to the lower + sign to change for a 1.

After dealing with the local resolution of overflow, now we turn to the global actions depending on the outcome of the process. How do we recognize that every overflow has been resolved? If there are no + signs left. That is, at each

local processor the 'head' is over a 1 and not over a + sign. It is rather easy to encode with a causal neighborhood:

Cn:

{

$\langle p_1, \overline{o_n} \rangle, \langle p_1, \overline{q_6} \rangle, ..., \langle p_t, \overline{o_n} \rangle, \langle p_t, \overline{q_6} \rangle,$ #resolving overflow is in process

$\langle o_x, * \rangle, \langle o_x, 1 \rangle, ..., \langle o_z, * \rangle, \langle o_z, 1 \rangle$ #every 'head' reads a 1

}

Remark: there is no need to pay attention to the indices of o-s, even though they are atoms. For, there are $*$ indicating 'head'-s only at the proper places.

As the next step, if this causal neighborhood applies, is copying the contents of the local processors to the small memory units and than to the main memory without dividing 0-s does not differ significantly from the other direction, which was given in detail above, I do not describe a determined region for this rule. The next step can be indicated in many ways, changing the inner states of the local processors, or turning them "off" and including a new inner state for the main processor for global communication, or a combination of these. In addition it could be put under another synchronization process if it is desired.

The other case is when the overflows cannot be resolved. In that case the last memory square of the first processor carries a + sign.

Cn:

{

$\langle p_1, \overline{o_n} \rangle, \langle p_1, \overline{q_6} \rangle,$ #processor no. 1

$\langle p_i, o_k, k' \rangle, \langle o_k, * \rangle, \langle o_k, + \rangle$ #overflow

$\langle p_2, \overline{o_n} \rangle, \langle p_2, \overline{q_6} \rangle, ..., \langle p_t, \overline{o_n} \rangle, \langle p_t, \overline{q_6} \rangle,$ #resolving overflow is in progress

}

Here o_k is a concrete atom, but it could also be singled out by its position if it is required.

We will use the internal state q_{ERROR} to indicate the problem. As the problem is global, the main processor will have the same internal state:

Dr:

{

$\langle p_1, \overline{o_f} \rangle, \langle p_1, \overline{q_{ERROR}} \rangle,$ #error message from processor #1

$\langle g, \overline{q_{ERROR}} \rangle,$ #global error message

$\| : \langle p_i, o_k, k' \rangle, \langle o_k, * \rangle, \langle o_x, + \rangle : \|$

}

As the local processors do not have internal states and p_1 and g have internal states that do not have rules that include them in their causal neighborhoods, the device stops computing at this point.[41] This concludes the description of our example.

[41] If such a description is desired where the device does not stop but is not changed by any further steps, then a rule can be included where both the causal neighborhood and the determined region only contain $\langle g, \overline{q_{ERROR}} \rangle$.

This detailed example demonstrates how to represent features of parallel computing in the Computable Dynamical Systems framework. That is, in the example above there was a main processor with memory access units and several further local processors, global and local communication and information transfer between the processors, and synchronization of local processors. Again, as it was mentioned in the introduction of this section, it is not formal, mathematical proof, it is only a demonstration that makes plausible, hopefully in a convincing way, that indeed every feature of actual physically realizable parallel computing can be represented in this framework. A precise mathematical proof of this statement, or at least an example like the above in that fashion is one direction the Dissertation could be continued.

* * *

After this example, we have to return to Sieg's two questions in the Concluding remarks of his (2002):

> "Is there a natural subclass of Turing computable functions in which these neural nets lie?", and "Are there mental processes for whose representation this aspect [i.e. secondary neighborhoods and regions] of Gandy machines might be crucial?". These are important and most appealing questions. (p. 257)

As we saw earlier, secondary causal neighborhoods and determined regions had to be introduced to coordinate the new atoms in Conway's Game of Life in the case when it expands. However, De Pisapia in his (2000) shows that artificial neural networks can be described as Computable Dynamical Systems without using them. Similarly, in the example above, and plausibly in case of all current parallel computing devices, there is no need for secondary causal neighborhoods and determined regions. The reason for this is that these machines do not expand. Everything is finite and only the contents are changing, but the structure does not grow. Even if one imagines a parallel device where the processors are Turing Machines, i.e. their tapes expand, they would not need secondary neighborhoods and regions. For, even though the Turing Machines will expand "locally," there will be no scenario where one Turing Machine's new atom, i.e. its expanding tape, overlaps with another one.

As parallel computing is also used to represent cognitive or mental processes, this example is relevant for the second question of Sieg. It possibly also suggests a modification of the first question. Namely, is there something special about Conway's Game of Life, or is there an interesting subclass of computable functions that do require secondary causal neighborhoods and determined regions? For, it seems that many models of computing are representable without secondary neighborhoods and regions.

4.3 Exploring the Borders

In this section models or parallel computations were examined. I showed that the various features of classical models of parallel computing, such as network,

PRAM, and BSP models, can be shown to be Computable Dynamical Systems. The aim of this subsection is to provide a broader spectrum of models of (parallel) machines that are not captured by the above examples, and, further, models which might be outside the scope of Computable Dynamical Systems.

Further Features of Models of Realistic Parallel Machines

The theoretical models of parallel computing discussed above now belong to the classical literature. There are, however, more involved processes in current actual parallel computing devices. For example, in the procedure above, the operations of local processors were never intertwined. That is, every local processor had its own memory access unit in the main processor, and as a consequence they were never reading from or writing to the same memory location and their reads and writes took the same amount of time (after synchronization). Thus, the procedure above satisfies what is now called *sequential consistency* after Lamport's (1979): "the result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program." (p. 690) However, to improve the performance of actual distributed systems, the sequential consistency requirement is usually relaxed in some way or another, leading to so called *relaxed memory consistency models*. Several features of such devices were not displayed in the example above, such as buffers and caches appearing on the hardware/software level, and non-determinacy on the computational level.[42] As the framework of Computable Dynamical Systems can accommodate not only programs (software), but the hardware itself, dealing with buffers and caches should not be a problem. Treating the possible non-determinacy of the output from different program runs might be more involved, because computations are deterministic and always take the same amount of steps in particular Computable Dynamical Systems. One possible solution seems to be to model the latency and speed changes of program runs in actual computing devices as exogenous effects inhibiting some computational steps from happening. See Section 5 for the treatment of exogenous sources.

Analogue and Quantum Computing

Gandy drew the following distinction while analyzing machine computability in his (1980):

> I shall distinguish between 'mechanical devices' and 'physical devices' and consider only the former. (p. 126)

By excluding 'physical devices' Gandy's aim was to "exclude from consideration devices which are *essentially* analogue machines." (p. 125) That is, to exclude analogue computing devices that obey the rules of classical physics on the one hand, and quantum computing devices on the other.

[42] Non-determinacy can occur even under sequential consistency models, as processors might execute their threads in different orders while still remaining sequentially consistent.

However, based on an unpublished draft from Gandy, it is clear that he believed that these computing devices cannot compute anything that is not Turing computable. His draft is quoted in Copeland and Shagrir's (2007):

> A number of examples have been given of physical systems (both classical and quantum mechanical) which when provided with a (continuously variable) computable input will give a non-computable output. It has been suggested that such systems might allow one to design analogue machines which would calculate the values of some number-theoretic non-computable function. Analyses of the examples show that the suggestion is wrong. [...] I claim that given a reasonable definition of 'analogue machine' it will always be wrong. The claim is to be read not so much as a dogmatic assertion, but rather as a challenge. (p. 221)

Of course, just because these devices do not compute anything that is not Turing computable, it does not mean that they fall within the scope of Gandy's and Sieg's analysis. This is similar to the case of Conway's Game of Life, which was known to be Turing computable, yet provided a partial reason for Gandy to include "parallelism" as a feature of machine computing.

As there is no generally accepted formal model of analogue computing (not even of computing devices) under the assumption of classical physics, I will rely on Cotogno's (2003) survey article on the computation (and possible hypercomputation) of physical systems. With respect to analog devices, it seems that the majority of scholars is in agreement with Gandy's claim. According to Cotogno, "[t]he most common view is that discrete approximations with any desired degree of accuracy can perfectly simulate computations on real-numbered quantities" and "analog computation is routinely simulated by digital means" (p. 199). Furthermore, that even those who claim that certain theoretical analogue machines can transcend Turing computability "admit that this superiority is merely hypothetical, and 'almost certainly unphysical"' (p. 200). Let us now turn to quantum computing.

In quantum computing a quantum bit or qubit is represented by a superposition of vectors in an orthonormal basis. More precisely, an orthonormal basis of unit vectors is fixed, which is usually denoted by $|0\rangle$ and $|1\rangle$, standing for the vectors $|\uparrow\rangle$ and $|\rightarrow\rangle$.[43] Then a qubit is represented as a unit vector in that basis, i.e. $\alpha\,|0\rangle + \beta\,|1\rangle$, where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. Quantum theory postulates that measuring devices have two preferred states represented by two orthogonal unit vectors, say $|u\rangle$ and $|u\rangle^{\perp}$. The "[m]easurement of a state transforms the state into one of the measuring device's basis vectors $|u\rangle$ or $|u\rangle^{\perp}$. The probability that the state is measured as basis vector $|u\rangle$ is the square of the magnitude of the amplitude of the component of the state in the direction of the basis vector $|u\rangle$." (Rieffel and Polak 2011, p. 16) For example, if the state of the qubit can be represented in this basis as $a\,|u\rangle + b\,|u\rangle^{\perp}$, then it will be measured as $|u\rangle$ with probability $|a|^2$ and

[43] This basis is called the standard basis, but any orthonormal basis would suffice.

as $|u\rangle^{\perp}$ with probability $|b|^2$. As a consequence, "[i]nformation about a quantum bit can be obtained only by measurement, and any measurement results in one of only two states, the two basis states associated with the measuring device; thus, a single measurement yields at most a single classical bit of information." (p. 17)

A phenomenon of quantum computing that most likely cannot be captured by Computable Dynamical Systems is quantum entanglement. This term describes a pair or group of particles in which the quantum state of each particle in the system is strongly correlated with the others, such that no independent description of each particle's state can be given. It means that two entangled particles can remain perfectly correlated even when separated by an enormous distance. It seems likely that this phenomenon cannot be accommodated in the framework of Computable Dynamical Systems, as those entangled particles should be treated as one causal neighborhood, but they can be at any distance from each other, so they violate locality and cannot be immediately recognized by the device. Sometimes this correlation is also interpreted as "action at a distance" which is faster than light, when the entangled particles are at enormous distances from each other. I cannot judge the plausibility of such interpretations, but they are definitely in conflict with Gandy's second physical principle, stating that "the finite velocity of propagation of effects and signals: contemporary physics rejects the possibility of instantaneous action at a distance." (Gandy 1980, p. 135)

At the same time the computational power of "standard" quantum computational devices is the same as that of Turing machines: "quantum computers can compute the same class of functions as is computable by a Turing machine."[44] (Nielsen and Chuang 2000, p. 126) They are believed to be significantly faster in case of certain problems, but they are not capable of computing functions that are not Turing computable.

Hypercomputing

Hypercomputing refers to models of computation that supposedly go beyond Turing computability. These models will be mentioned here rather briefly, though they are not necessarily related to machine computation. I will mention two particular models of hypercomputation, a rather famous one and one that was presented as a response to Gandy's challenge. Then I will turn to Cotogno's (2003) for some general comments.

Probably the most well-known hypercomputational model is Siegelmann's

[44]The quote continues: "The difference between quantum computers and Turing machines turns out to lie in the *efficiency* with which the computation of the function may be performed – there are functions which can be computed much more efficiently on a quantum computer than is believed to be possible with a classical computing device such as a Turing machine." It is suspected that the class of effectively computable functions by quantum computers, BQP (Bounded error, Quantum, Polynomial time), is a strict superset of both classes P and BPP (Bounded error, Probabilistic, Polynomial time); however, it is not proven. On quantum complexity theory see Bernstein and Vazirani's classic (1997).

analogue recurrent neural network (1999).[45] This model is an analogue computing device that computes in parallel, and it "goes beyond the Turing limit". Siegelmann shows that when the internal weights of these neural networks are set to be arbitrary real numbers, they can compute functions that are not computable by Turing machines. For example they can recognize all binary languages, not just the computable ones. Siegelmann considers her neural nets as idealized mathematical models, so "the precise real values do not constitute an unreasonable constraint" on them. (p. 20) Clearly, arbitrary real weights cannot be accommodated in the framework of Computable Dynamical Systems, as it would violate the finiteness principle. Martin Davis criticizes this aspect of Siegelmann's model: "Since the non-computability that Siegelmann gets from her neural nets is nothing more than the non-computability she has built into them [via real numbers], it is difficult to see in what sense she can claim to have gone 'beyond the Turing limit'." (2004, p. 203)

Gandy's "challenge," as quoted above, was taken up by Kieu in his (2002) based on his ideas for quantum hypercomputing. However, Kieu's work is criticized by Piccinini in a rather similar fashion to Davis' criticisms of Siegelmann and Copeland: "Kieu's scheme does not appear to be workable. For one thing, it requires infinite precision in setting up and maintaining the system (Hodges 2005). For another thing, Kieu does not provide a successful criterion for knowing when the system has evolved into the solution state". Thus, Kieu's model, requiring infinite precision, violates the same principles of Computable Dynamical Systems as Siegelmann's.

Cotogno in his (2003) surveys several kinds of hypercomputational models, such as models of infinite computation and supertasks, interactive computing, and analogue and quantum computing. Concerning the latter ones he points to the exploitation of infinite precision just as Davis and Piccinini did. About infinite computation and supertasks he says "that classical and relativistic physics allow us to think of computers capable of actually infinite operations in principle, but only within idealized conditions, and with *ad hoc* exclusion of quantum and thermodynamical constraints." All in all he concludes "that non-conventional approaches to computation are but projects to obtain ever-increasing degrees of efficiency, but do not affect computability *in principle*: the functions that can be effectively evaluated are nowhere external to Turing definability." (p. 215)

[45] Although Siegelmann's model is motivated by neural networks and not by machines, it is discussed here as it is quite different than the cognitive models in the next section, and, more importantly, it is proposed as a theoretical model of hypercomputing.

5 Models of Cognition as Computable Dynamical Systems

The aim of this chapter is to show that most of our contemporary cognitive models are computable. It is not a surprising statement in itself as many projects in the field of cognitive science have a leaning towards computability in their core. The main aim here is to demonstrate that these models can be represented as Computable Dynamical Systems. Although in the previous chapter it was shown how parallel computing can be represented in the framework of Computable Dynamical Systems and it was mentioned that De Pisapia showed earlier how to represent artificial neural nets, this task would be almost inexhaustible if the further cognitive models would have to be covered one by one. Instead, in this chapter I will rely on David Danks' claim in his *Unifying the Mind* (2014) "that multiple cognitive processes can all be understood as distinct, though related, operations on a common store of cognitive representation structured as graphical models." (p. 4) Thus, based on Danks' unification, my aim is to show how to represent graphical models and certain operations thereon in the computable dynamical systems framework. Before turning to the details of graphical models and the particular operations thereon I would like to treat, I am going to summarize Danks' general view in a bit more detail.

As already mentioned above, Danks proposes a novel cognitive architecture and his main claim is "that multiple cognitive processes can all be understood as [...] operations on a common store of cognitive representations structured as graphical models." (p. 5) Speaking of a new cognitive architecture raises questions about its empirical support, its scope, and relation to previous theories. Danks' architecture gains its empirical support, and in some sense, its scope through its relation to previous cognitive theories:

> Very few theories in cognitive science are expressed in terms of graphical models, and so a major part of my effort will involve showing that multiple cognitive theories can actually be fruitfully expressed in the graphical models framework. That is, these theories can be (re)interpreted as operations on graphical models, though this is not the language that has been used by the cognitive scientists who developed the theories. [...] The key point, though, is that we can translate all these theories into the graphical models framework, and so we can understand much of our cognition as operations on graphical models. (pp. 6-7)

The "reinterpretation" of cognitive theories in this framework also guarantees the empirical support for Danks' architecture. "That is, if an empirically well-supported theory can be expressed using graphical models, then we immediately know that the graphical models framework can account for those empirical data." (p. 7) Thus, the graphical model "inherits" the empirical support of the original cognitive theory.

According to Danks this approach explains two different aspects of cognition that are rarely studied, let alone together. These two features are the "relatively

seamless way in which we shift between seemingly distinct cognitive operations; and our ability to (correctly) attend precisely to the information, features, and possibilities that are relevant to our goals" (pp. 2-3). Where by the seamless shifting between distinct cognitive tasks the following experience is being referred to:

> We move smoothly between categorizing objects in the world, making inferences about them, using those inferences to reason about the causal structure of the world, making plans given that causal structure, taking actions based on the plan, observing and categorizing the outcomes of our actions, learning from those outcomes, and so on. (pp. 10-11)

The seamless shift between the operations[46] in this framework is explained exactly by the different operations acting on the "shared representational store of graphical models." The other feature, attending to the relevant information during our cognitive processes, is a consequence of graphical models directly representing notions of relevance. (p. 4)

However, Danks emphasizes that he does not claim this framework to be universal. That is, it is not claimed that every aspect of cognition can be explained in terms of operations on graphical models. Even more strongly, Danks believes that causal perception "produces cognitive representations that are almost certainly *not* graphical models" (p. 7).

Danks says that his approach "is unbashedly computational":

> [C]ognition will be understood as precise operations on mathematically well-specified objects. More precisely, a background assumption of this work is that it is appropriate to think about aspects of cognition as fundamentally involving learning, transforming, making inferences using, and manipulating structured representations about the world. (p. 13)

Which, of course, fits well with the framework of computable dynamical systems.

The computational approach does not settle in itself the commitments of the proposed theory. Danks explains the commitments of his theory as follows:

> This account is committed to the realism of these representations [as graphical models] but is largely agnostic with regard to realism about the processes (though there must be suitable processes that can use the information encoded in the representations). More specifically, the account proposes that there are persistent objects in the mind that subserve a wide range of cognitive processes, but where the precise processing method might, but need not, be identical in all domains or contexts. (p. 36)

[46] Which in many cases cannot even be sharply differentiated from each other, that is, it is not always clear when one operation ends and another begins.

He adds later that this representation realism "requires only that people behave in systematic (and systematically interpretable) ways that are best explained as operations on graphical models" which commits him to "realism about representations that is entirely consistent with them being neurally distributed or emergent, as long as the distributed representations are appropriately *stable* across tasks and environments." (p. 183)

After explaining the general approach and the commitments of the architecture, Danks turns to reinterpreting cognitive theories in terms of graphical models. The reinterpretation is done in two steps. First it is shown how to represent the observations and experience of the agent in terms of graphical models. Then it has to be shown how the operations of different cognitive processes can be carried out on graphical models. As Danks claims the unification of representations only, the second task can appear to be almost inexhaustible. However, as most of the cognitive processes are already formally and computationally specified, it is enough to indicate how those operations could be carried out on graphical models once their input is put into that framework.

The first area that is reinterpreted in such a way is causal cognition. Where "causal cognition can be broadly divided into causal learning and causal reasoning, where causal learning is (roughly) the acquisition of new causal beliefs or changes to existing ones, and causal reasoning is the use of those beliefs in various ways." (p. 67) Causal learning is further divided into causal inference and causal perception. Causal inference is creating our representations of causal structures "from a set of cases none of which explicitly convey causal information." (p. 69) On the other hand, it is causal perception that "produces cognitive representations that are almost certainly *not* graphical models" (p. 7). Causal reasoning is the use of our representations of causal structures when predicting or inferring something. Danks remarks that unifying causal cognition in terms of graphical models is "perhaps unsurprising" as, on the one hand, it is the area of cognitive science that uses graphical models most extensively, and, on the other hand, some kinds of graphical models were particularly motivated by and developed to capture causal relations. (p. 98)

The next area of cognition reinterpreted by Danks in terms of graphical models is categorization through concepts:

> Concepts are used throughout our cognition to carve up the world, make inferences about new individuals or objects, organize our knowledge about diverse sets, and perform many other functions. It is natural to think of them as the foundational elements or building blocks for almost any cognitive representation. (p. 99)

He shows how to put the major theories of concept representation and categorization, such as exemplar, prototype and the so called theory theory accounts, into his cognitive architecture. Then, again, as in the case of causal cognition, the next task is to show how to reason and argue with concepts.

> In the graphical models framework, the problem of categorization is essentially one of a model choice: given a set of possible graphical

> models and some new individual, what is the likelihood that each graphical model could have produced that individual? (p. 115)

Both cases, representing concepts and reasoning with them, involve only one concept at a time. Danks also shows how to treat between-concept relations and hierarchies of concepts, which involve multiple concepts at the same time. That is, those cases when an individual falls under several concepts, such as a particular animal falling under both concepts of 'dog' and 'mammal' as well as several concepts, such as 'dog' and 'cat' falling under another concept, e.g. 'mammal'. This results in building webs from graphical models representing single concepts.

The last big area Danks considers is decision making. He remarks that both causal cognition and treating concepts were focused on representations (and reasoning with them), but that they are "essentially impotent on their own; they are only useful if they are connected in some way with decision-making processes that can use them." (p. 129) Here he shows first how traditional models of decision making, such as the 'take the best' heuristic, can be implemented on graphical models as inputs. He also proposes novel decision making models on graphical models as inputs, where the agent's selection of (their) possible actions or interventions are treated as operations on graphical representations. Finally, he introduces decision networks. The aim of these decision networks is to capture the *process* of decision making by graphical models; in the two earlier cases, graphical models were mere inputs for the processes. The decision networks are richer graphical models than the ones used in the previous cognitive tasks. Here four different kinds of nodes are used to encode "actions, values, factors in the world and the decision-making method itself." (pp. 142-143) Both the usual graphical models and the decision networks will be described in the next subsection.

Then, finally, Danks argues for the unification of these pieces and for his commitment to a realism towards representation in terms of graphical models. He emphasizes that his claim is not the simplistic claim that the "shared mathematical description" of these cognitive representations and processes entail some kind of "metaphysical identity." Instead:

> The arguments [...] for the existence of a shared representational store–that is, some measure of metaphysical identity–is that it provides the best explanation for the range of empirical data [...], principally about what we might call "cross-cognition transfer." (p. 152)

The remaining part of the book is devoted to questions such as how this cognitive architecture could be further tested empirically. What its relation is to alternative approaches, what implications it has for the notion of modularity, and what the challenges, open questions and next steps are for this novel cognitive architecture.

5.1 Representing Unified Cognition in the Framework of Computable Dynamical Systems

As Danks was quoted above, the main claim behind his unification of the mind is "that multiple cognitive processes can all be understood as [...] operations on a common store of cognitive representations structured as graphical models." (p. 5) Thus, in order to put Danks' ideas into the framework of computable dynamical systems, it has to be shown how to represent: (1) graphical models and (2) cognitive processes understood as operations on graphical models.

First I will show how to represent graphical models as hereditarily finite sets. Then, to accommodate Danks' commitment to a kind of realism towards representations as graphical models more directly, the computational models in this chapter will operate directly on graphs. As I mentioned above, there are several different cognitive processes for which Danks indicates how they can be carried out on graphical models. Describing all these processes in detail within the computational dynamical systems framework would take up another book. However, most of these processes are already formally and computationally specified in their original form; this is why it was enough to indicate how they could be implemented in the framework of graphical models. As a consequence I will not describe all of them in detail either, as Computable Dynamical Systems, but only provide a few examples and focus on those models which are heuristic, that do not exploit the vast computational powers of computers available today, but are designed to closely match human behavior to the extent to even reproduce its biases. These models will come from the area of causal structure learning, that is, when agents build up and modify their representations in terms of graphical models about causal structures they observe.

Before turning to the formal details, a similar methodological remark as the one in the previous chapter is in order. Namely, the demonstration of the representation of several cognitive processes in the framework of Computable Dynamical Systems does not constitute a representation theorem. For, many features of these cognitive models are not specified. For example there is no characterization of possible or admissible operations, or a precise definition of locality, not even in case of particular cognitive processes. Hence, the demonstration below which makes plausible that all features of cognitive models can be represented in this framework is the best that can be given as of now. This also calls for the characterization of such issues in the future. Now we turn towards directed acyclic graphs and cognitive processes represented in terms of them.

Graphical Models as Computable Dynamical Systems

Graphical models have two components, a graph component and qualitative component. First I will discuss the graph component, and then the qualitative component and its connection to the graph component. Finally some of the operations on graphical models will be discussed.

In general, if the graphical model deals with n variables, then the graph

component will have n vertices. The graph component can be undirected or directed, usually the directed graphs are required to be acyclic, i.e. there is no closed path in the graph on which every edge is directed in the same direction. These graphs are called DAG-s, standing for directed acyclic graph. The standard mathematical treatment of graphs is to represent them as finite sets, i.e. they are already represented as hereditarily finite sets. A graph is understood as $G = (V, E)$, where V is the finite set of vertices and E is the finite set of edges, where the edges are unordered pairs of vertices, namely $E \subseteq V \times V$. Directed graphs are denoted as $\overrightarrow{G} = (V, A)$, where A stands for the directed edges or arrows, which are represented as ordered pairs of vertices.

To accommodate the different kinds qualitative components, the following, slightly modified graphs have to be treated as well: (1) graphs that have weights on their edges; (2) graphs whose vertices can have different properties or belong to different types. In case of (1), weights assigned to edges can be encoded with ordered triples, thus $u \xrightarrow{6.23} v$ appears as $\langle u, v, 6.23\rangle$.[47] In case of (2) we assume that there is a finite set of properties or types of vertices. Properties can be encoded with the technique that was used as one of the conventions in Section 4. Namely, that these properties are encoded not as bare atoms, but as r^k where r is an atom an k is a fixed integer. In this way nodes with their properties can be encoded as ordered pairs and will be easily distinguishable from directed edges, as those are ordered pairs of two atoms. This concludes the graph part of graphical models and now we turn to the qualitative component.

In most graphical models Danks considers the qualitative component is fully characterized by a joint probability distribution or density function of the variables. The variables can be discrete or continuous. Here I only discuss the discrete case, the so called Bayesian networks.[48] To establish a rather strong connection between the graphical and the quantitative component, usually two conditions are imposed on these kinds of graphical models. The Markov condition, informally, requires that those variables which are not adjacent in the graph are independent (possibly conditionally on some set of variables) in the quantitative component. The second, so called Faithfulness condition requires, informally, that the only independencies in the network are those implied by the Markov condition. If these conditions are met, then the joint distribution can be factorized into n conditional distributions, taking the following form:

$$P(\overline{V} = \overline{v}) = \prod_{i=1}^{n} P(V_i = v_i \mid \overline{Par_i} = \overline{par_i})$$

where $\overline{V}$ stands for all the variables $\{V_1, ..., V_n\}$ and $\overline{Par_i}$ denotes the set of graphical parents of V_i. This is called the Markov factorization of the model.

[47] Here it is assumed that there is an encoding fixed for the possible numbers in terms hereditarily finite sets.

[48] In the continuous case we can assume that, just like when they are computed with digital computers, discrete approximations and approximation techniques are used. Another possibility is to use computable reals as De Pisapia did, for details see his (2000, pp. 57-59).

Recovering the causal structure from the covariational data has an extensive literature. The task is to find algorithms that factor the available finite numerical data into the above form through computable numerical calculations; once the factorization is available it is straightforward to construct the graphical component. As all the proposed methods are algorithmic and operate on finite numerical data, we can assume that they are computable by Computable Dynamical Systems as well, since in the previous section it was shown how realistic computing devices can be treated in that framework. Once a graphical model is available, its Markov factorization makes computations with the graphical model rather efficient. Those operations can easily be captured by Computable Dynamical Systems: as the distribution of a variable depends only on its parents, the relevant nodes can be collected into the causal neighborhoods of the System.

Once such a fully specified graphical model is available, several further tasks become easily solvable. For example, the unconditional and conditional dependencies and independences can easily be read off from the graphical component, using d-separation (see p 52. in Danks' (2014)). This graph algorithm checks certain properties of paths within nodes in the graph and is clearly computable, and it is easy to handle with operations of Computable Dynamical Systems. Another frequent use of graphical models is to treat interventions. The case when a variable V_i is intervened on, say it is set to 1, can be captured by the following modified graphical model: remove the edges between V_i and its graphical parents Par_i and set the distribution of the node to $P(V_i = 1) = 1$. This can also easily be achieved with Computable Dynamical Systems.

Danks shows in his (2014) that many cognitive problems can be reinterpreted in this framework as a task of discovering the graph component or estimating its parameters and so on. Furthermore, they can also account for actions of agents as well, assuming that other cognitive processes have access to the graphical representations. For example, take the task to bring about a certain outcome in the real world, say, to turn on the lights. It can be understood as the following, somewhat simple, problem on a graphical model. On which variable should we intervene on to set the variable responsible for "lights" to value 1 if we have the causal structure available for us as a graphical model? The solution in terms of the graph can be represented as tracing back paths from the variable of interest until we find another variable we can intervene on, say "turn switch". Thus, in this simplified case, human causal reasoning can be turned into a, clearly computable, graph search problem.

Actions of agents in most cases are less straightforward than in the above example. Decision networks were introduced to capture the more complex structure of the different factors that are relevant for decision making. These factors, such as actions, values, factors in the world, and decision making method, are encoded by different types of nodes, as visually indicated in Figure 6, by rectangles, diamonds, ovals and triangles respectively.

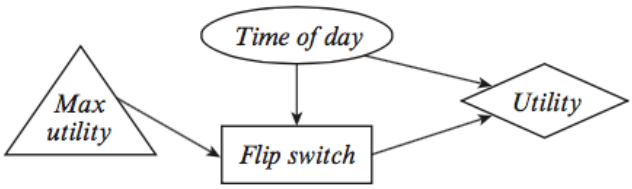


Figure 6, from (Danks 2014, p. 143)

It was indicated in the discussion of the graphical component above how to encode different types of vertices.

The exogenous variables, such as the "time of the day" in the example can be handled as follows. Certain atoms can receive input from exogenous sources at any time during the computation and their content cannot be overwritten by the system. In Section 5.3 below where a detailed treatment of heuristic causal structure learning is given, the "observational field" will be such an exogenous source. Thus, again, the 'new' features of the decision networks can all be handled by Computable Dynamical Systems.

One feature of graphical models is left to be discussed here: randomness. Graphical models are used to generate random data, reproduce decisions of human agents that seem to have some level of variety instead of being deterministically determined and so on. However, in all cases when randomness is produced or reproduced by such models, 'randomness' stands for semi-random sequence generated by computable functions. Computable Dynamical Systems can incorporate that as well, by containing a Turing machine that produces such a random sequence as a substructure, or if randomness is 'local', such a Turing machine can be placed on every node, where 'random noise' needs to be generated etc. This concludes this subsection on DAG based graphical models as Computable Dynamical Systems.

5.2 Causal Structure Learning

Learning causal structure and representing it as a DAG has a long-standing literature in the field of machine learning and statistics, see for example (Pearl 1988) or (Spirtes, Glymour and Scheines 1993/2000). This approach usually relies on purely covariational data. However, we know that human agents take into account other causal cues as well, such as the use of interventions and temporal order of the events they observe. When only covariational data is available to human learners, which is very rare in realistic situations, on average they do rather poorly, only slightly better than chance. Moreover, in many cases, when available, they rely on cues more heavily than on covariation. That is, they let these cues override covariational data when they conflict (Lagnado and Sloman 2004 and 2006). It is, at least in part, because:

> [I]nferring possible causal models in a purely data-driven fashion involves a significant computational load. Although this may be manageable by a powerful computer, it is less likely to be achievable

> by humans with limited processing and memory resources. (Lagnado et al 2007, p. 158)

Of course this approach was not intended to capture actual human behavior. As Waldmann and Martignon put it:

> Such models [e.g. Pearl 1988; Spirtes, Glymour, and Scheines 1993] are typically developed as normative tools for statistical analysis, and they often aim at developing strategies to bootstrap causal structures from covariation data in a bottom-up fashion. These methods are not intended to model everyday causal reasoning. On the contrary, they are often motivated by the assumption that causal analysis need to be guided by expert systems that embody Bayesian strategies. In our view, it is unlikely that human learners are good at inducing the causal relations between several interconnected events solely on the basis of covariation information. (Waldmann and Martignon 1998, p. 1103)

Thus, we see that for capturing human cognition, this approach to causal learning is not the most suitable. The other approach is to create heuristic models of human causal structure learning. Contrary to what connotations the label 'heuristic' might has, this approach also produces algorithmic descriptions of human causal structure learning. 'Heuristic' here means that probabilities and other numerical indicators are not directly represented and calculated with, instead the goal is to find simple rules of thumb that the learners follow. Hence, these heuristics can be turned into production rules. The aim of this approach is to provide a realistic picture of human causal structure learning, taking into consideration the limited processing and memory resources of humans and even reproducing its biases and errors. These models usually assume that agents have only one (or very few) current hypothesis about the causal structure which they represent as a graphical model and update it locally (e.g. edge by edge).

In the following sections the interest will mostly be restricted to those models of causal structure learning which do not involve numerical calculations, most prominently (Fernbach and Sloman 2009) and (Lagnado and Sloman 2006). It means that I will treat such models of causal structure learning as the Local Prediction-error Learning model of Wellen and Danks (2012) only as short examples. There agents update their only hypothesis graph edge by edge, using causal strength estimates which are numerically calculated. Their model also fits the experimental data Lagnado and Sloman's (2006).

In the next subsection I summarize those heuristic models whose intention is to explain human behavior in causal structure learning. These models take interventions and temporal order of the events into account beyond pure covariational data, and do not rely on numerical calculations.

Heuristic Models of Human Causal Structure Learning

In this section I survey the two papers I am going to rely on about causal structure learning. That is, Fernbach and Sloman's heuristic model of causal

learning when interventional data is available for agents. And Lagnado and Sloman's observations about causal learning when agents observed the temporal order of the events.

Fernbach and Sloman's heuristic model in case of available interventional data

Fernbach and Sloman proposed a heuristic model of causal structure learning in their (2009). The model is based on local computations. Where locality is not defined strictly but is somewhat clarified by the following descriptions. "When faced with a complex learning problem involving several variables, people break the problem up by focusing on evidence for individual causal relations rather than evidence for fully specified causal structures." (p. 680) This is an important aspect of their model, even in the case of small causal structures investigated in their experiments with only three variables. They assume that structural inferences are "restricted to parts of the structure that were active or informative on a given" case. (p. 681) That is, as we will see later, in most cases it is assumed that the participants in the experiments pay attention only to those variables, that are currently activated. It is important to emphasize however that there is no restriction on the number of the links that can be learned from a given observation, as "participants sometimes made inferences about multiple links simultaneously, as when two effects of a common cause were simultaneously present." (p. 681)

As the aim of the model is to achieve a level of psychological fidelity, it makes minimal demands on memory, attention and computational power. Minimal demands on memory manifests in two ways. First, "causal structure is inferred from a series of observations by serially updating a single hypothesized model" (p. 680). That is, the agent does not have to remember several hypothesis structures and update all of them from observation to observation. The second is that learning is based on small data sets; in the experiments participants are presented only with five observations of a given causal structure. Fernbach and Sloman remark that "participants observed small data sets, insufficient for reliably estimating causal strength, and still made systematic structural inferences." It led to "systematic inference of extraneous causal links," a systematic bias that is predicted by their heuristic model but not by Bayesian ones. Reproducing this bias on small data sets reinforces the psychological fidelity of their model.

Beyond pure covariational data, interventions served as further causal cues for the participants. They were presented with five observations, from now on, trials, generated by either a chain or a common cause structure. See Figure 7 below.

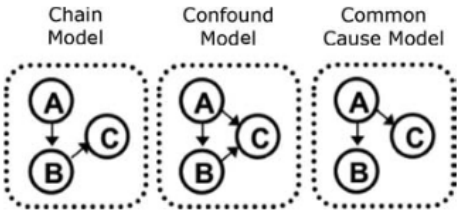


Figure 7, from (Fernbach and Sloman 2009, p. 681)

The causal links work effectively 80% of the time, but the participants were not aware of this information. Participants were not choosing the interventions for themselves, only observed them. The variable that was intervened on was clearly identified. "[D]uring the interventions, there was no lag between the intervened-on and affect variables" (p. 683).[49] After observing five trials, participants had to select the causal structure on a screen.

The initial heuristic model based on local computations proposed by Fernbach and Sloman is the following: "a local computation considers only one hypothesis at a time. The hypothesis is constituted by the union of all causal links implied by local cues on individual trials." (p. 681) That is, it is assumed that participants start with an empty structure as a hypothesis. A causal link is added between the intervened-on variable and another variable, if the variable which is not intervened on is also activated in the same trial. As the hypothesis is the union of all causal links implied during the trials, it means that once a causal link is added to the hypothesis, it will not be removed later. Which leads to the following bias in learning:

> The learner may not realize that a simpler hypothesis is also consistent with the data because only one hypothesis is considered. (p. 681)

More concretely it leads to the following bias. Assume that the interventions a participant is observing are generated by a chain structure where $A \to B \to C$ just like in Figure 7. Based on the heuristic, if there is an intervention on A which activates both B and C, both edges $A \to B$ and $A \to C$ are added to the current hypothesis (if not already included). If in another trial an intervention is observed on B which activates C, then the edge $B \to C$ is added to the hypothesis; this configuration is called confound structure, as shown in Figure 7. Based on the heuristic, no simpler hypothesis is ever tested against this confound structure. It is important to emphasize that this confound structure is consistent with the data, however it is more likely to be generated by a chain.

[49] Fernbach and Sloman claim that "interventions provided *implicit temporal information*, namely, that any active sliders must have moved after the intervened-on slider." (italics by me, p. 681) However, this information (based on previous knowledge) is rather restricted in the sense that if the intervened-on variable effects both other variables, participants do not learn anything about their temporal order. When I will discuss those experiments which explicitly involve temporal order, this case will be consistent with the case of all effects appearing at the same time.

Thus, in case of chain generated trials the heuristic model would predict the confound structure as the choice of the participants, while Bayesian models would predict chains.

It is also possible that during five randomly generated interventions no intervention on A is observed. For the heuristic model it indicates only either 0 or 1 edge in the structure, as it cannot infer a link for which no evidence was seen. In contrast a Bayesian model, depending on its parameters, might includes some.

The results of the experiment are shown below in Figure 8.

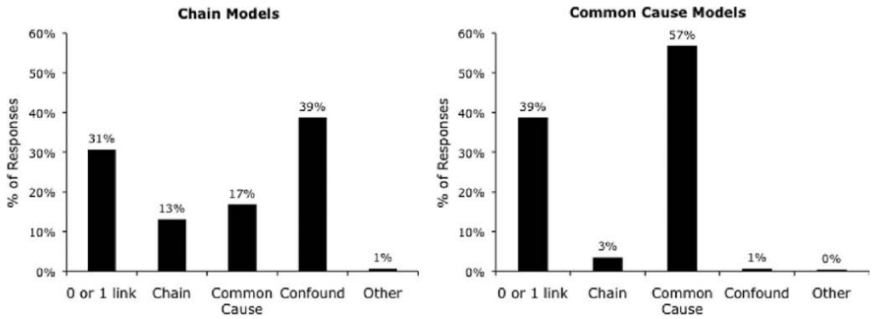


Figure 8, from (Fernbach and Sloman 2009, p. 684)

Fernbach and Sloman describe the results as follows. "When the generative model was a chain, confound models and zero- and one-link models were predominant. When generative model was a common cause, common cause models and zero- and one-link models were predominant." Which fits well the heuristic model, that predicts that "participants should have difficulty inferring chains but should have no trouble inferring common causes given insufficient data." (p. 683) And the data indeed conforms to the bias predicted by the heuristic model for the preference of the confound model when the generative structure is a chain.

The performance of the heuristic model was then compared to two Bayesian models. While the heuristic model predicted 87% of overall responses correctly, none of the Bayesian models did better than 65%.

Although participants were likely to infer a confound structure when the data was generated by a chain, during the five trials they observed, it was unlikely that they would observe a trial which would allow them to clearly distinguish between chains and confound structures. In Experiments 2 and 3 Fernbach and Sloman tested whether observing such "diagnostic" interventions would make a difference in participants' responses.

We will call a case a "diagnostic" intervention when A is intervened on, it activates B, but the $B \rightarrow C$ causal link fails, and C stays inactive. As in the experiments the probability of a causal link working successfully was 80%, the probability that a diagnostic intervention is seen under chain and confound structures is the following:

$$P(\textit{diagnostic intervention} \mid \textit{chain structure intervened on } A) =$$
$$= P(A \to B \textit{ works}) \times P(B \to C \textit{ fails}) =$$
$$= 0.8 \times (1 - 0.8) = 0.8 \times 0.2 = 0.16$$

while

$$P(\textit{diagnostic intervention} \mid \textit{confound structure intervened on } A) =$$
$$= P(A \to B \textit{ works}) \times P(A \to C \textit{ fails}) \times P(B \to C \textit{ fails}) =$$
$$= 0.8 \times (1 - 0.8) \times (1 - 0.8) = 0.8 \times 0.2 \times 0.2 = 0.032.$$

We can see that the diagnostic intervention is way more likely to appear if it was generated by a chain structure. However, even in that case its probability is only 16%, hence in most cases participants did not observe them during the five trials.

In Experiment 2 it was made sure that participants do see a diagnostic intervention. The participants were divided into two groups, called "Early Group" and "Late Group." Participants in the Early Group observed a diagnostic intervention in the first trial, while participants in the "Late Group" as the fourth out of five trials. According to the heuristic model, participants in the Early Group are not yet committed to any causal relations in the structure and see the diagnostic intervention only as evidence for the $A \to B$ link. On the other hand, when the diagnostic intervention comes later, when the learner's current hypothesis is already a confound structure, they might modify their hypothesis in face of surprising (but not inconsistent) data. In contrast, for Bayesian models there is no difference when the diagnostic intervention is observed.

The experimental data shows that the order in which the trials were presented clearly affected causal structure learning. At the same time, "[e]ven given the diagnostic intervention, which is highly unlikely to come from a confound model, participants still tended to infer confound models over chains across both conditions." According to Fernbach and Sloman, the fact that the late diagnostic intervention has an effect on learning the correct structure but not the early one "speaks to the automatic application of local heuristics." (p. 688) In this case, again, the heuristic model did fit the data better than any of the Bayesian models tested.

In Experiment 3 participants were trained about causal structures as chain and common cause structures. Participants were trained by different methods which did affect their choices. In one group the chain did become the dominant choice over the confound structure. Furthermore, the Bayesian model predicts the behavior of this group better then the heuristic one. In discussing the results of the last experiment, Fernbach and Sloman gave the following explanation:

> While the overall beneficial effect of training may not be surprising, what is surprising is that despite the priming of chain models, a subset of participants in each group still behaved like the participants in Experiment 1 and was fit well by the local computations heuristics. The results thus reveal when the heuristic model we propose is applicable and when it is not. Local computations are a default

> that can be overcome by effort or by an environment that provides support.
>
> The relative success of the Bayesian model when people were taught about chains shows that people were able to learn these simple three-variable causal structures when given a cue to consider one of those structures. The effect of this teaching could be modeled in a Bayesian framework as a change in people's prior probabilities over causal structures, a low prior for chains when they are not taught and a higher prior when they are. Although this is descriptively correct, we do not see that it provides any added explanatory value. (p. 691)

When putting the heuristic model into the framework of computable dynamical systems, I will also introduce a production rule which captures the learning effect from a diagnostic intervention, if the learner is currently committed a confound model. That is, I will reconstruct the untrained learners of Experiment 1 and 2 who use local heuristics.

Lagnado and Sloman's observations in case of available temporal order data

On the question of how temporal order influences causal structure learning I will rely on the data from Lagnado and Sloman's Study 1 from their (2006). They do not propose any concrete theory to generate or predict the data, as their main question was whether temporal cues contribute to causal learning beyond covariational data. They found that "temporal order can override sparse covariational information and lead to spurious causal inferences." (p. 453) As we will see, the data from their findings can be fit rather smoothly with a slightly modified version of the heuristic model proposed for learning from interventions by Fernbach and Sloman.

Although informative in many cases, observed temporal order as a causal cue can also be misleading, e.g. rooster does not cause the sun to come up. However, Lagnado and Sloman found that observed temporal order information does dominate causal structure learning over covariational data.

The scenario presented for the participants in their study was "inspired by viruses (electronic or otherwise) whose temporal order of transmission is not necessarily reflected by the order in which they manifest." (p. 452) Participants were inspecting a computer network which was effected by a computer virus. They observed the temporal order in which the computers crashed and were asked to recover the network structure based on that information. "Clearly the temporal order in which the e-mail virus manifests (by crashing the computer) does not guarantee an inference about the [causal] order in which the computers were infected or about who infected whom. More generally, the causal order in which events occur cannot simply be read off from the temporal order in which events occur." (p. 452)

To see how observed temporal order and covariational data are integrated while learning a causal structure, participants looked at a network where the

covariational information was kept constant (i.e. the generating causal structure was the same) but the observed temporal order was manipulated. The computer network that generated the covariational data looked as the one in Figure 9.

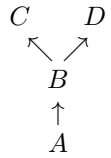


Figure 9

Participants were sending 100 test messages to computer "A" and then observed in each case the temporal order in which computers got affected. In this study participants were told that the connections between computers work 80% of the time and that not all computers are connected to each other. Manipulating the temporal order meant that for each participant during the 100 test messages the temporal order they observed was the same, but there were four different observable temporal order conditions, as shown in Figure 10 below.

Temporal Order of Display of Information (1-s delays) for the Four Conditions in Study 1

	Time steps			
Condition	t_1	t_2	t_3	t_4
1	A B or C or D			
2	A	B	D	C
3	A	D	C	B
4	A	B	C or D	

Note. Inclusive "or" is used throughout. For example, in Condition 4 at Time Step 3, either C or D or both can occur.

Figure 10, from (Lagnado and Sloman 2006, p. 454)

A "summary model choice" structure can be seen for each of these observable temporal order conditions in Figure 11. Only those edges are contained in these structures that were endorsed by at least 50% of the participants, and the thickness of the arrows correspond to the percentage of participants selecting that link, the thickest link being 100%.

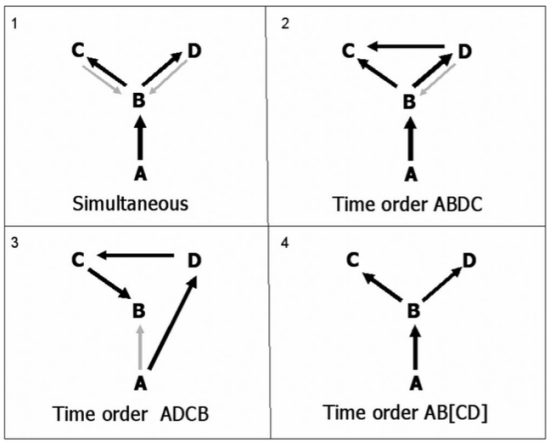


Figure 11, from (Lagnado and Sloman 2006, p. 454)

Lagnado and Sloman assess the results as follows:

> Inspection of these models shows use of both temporal and covariational information, with the former dominating the latter. Thus, even though all four problems had identical structure and therefore generated the same covariational data, link choices were heavily influenced by the temporal ordering. This is most apparent when the temporal ordering conflicted with the underlying structure (Conditions 2 and 3). In both conditions, participants inserted links that were implied by the temporal cues but not by the patterns of covariation. (p. 454)

As mentioned above, Lagnado and Sloman gave no concrete theory to predict this behavior. However, they provide some plausible explanation of how these patterns could have been arrived at. When talking about condition 2 in Figure 11 with observed temporal order (A, B, D, C), they explain the inferred structure as follows:

> A plausible explanation for this pattern of judgements is that participants first used the [observed] temporal ordering to hypothesize an initial model ($A \rightarrow B \rightarrow D \rightarrow C$). However, occasionally they saw a test pattern that contradicted this model ($A, B, C, \sim D$). To accommodate this new evidence, they added a link from B to C but did not remove the redundant link from D to C, because this still fit the [observed] temporal ordering. (p. 459)

As it will be shown, this description is in accord with the heuristic model proposed by Fernbach and Sloman. In the next subsection I show how to unify and put these models into the framework of computational dynamical systems. The production rules to include links in the hypothesis of the participant will be the same in both cases. The slight difference will be the way the input is received and put into the same form, so the exact same rules can be applied to them.

5.3 Description of Heuristic Models of Causal Structure Learning as Computable Dynamical Systems

In this section I unify and put the heuristic causal structure learning models into the framework of computable dynamical systems. First I will follow closely Fernbach and Sloman's model, where participants observed a system that was intervened on. Then I will turn towards Lagnado and Sloman's findings in the case where temporal order information was available instead of interventional data. Finally I will introduce production rules that mimick the behavior of the participants. Although both papers deal with small graphs, that is, sparse structures on three or four variables, the rules introduced below are applicable in general. That is, I do not exploit the small size of the structures and "hard code" the rules for all cases. Quite the contrary, the heuristic model can be captured by two rules applicable on graphs of any size. As the rules below are general, one can also construct hypotheses about human behavior in case of larger graphs based on them, as I will show.

At the beginning it should be pointed out that both the heuristic model proposed by Fernbach and Sloman and the explanation given by Lagnado and Sloman for the behavior of their participants are rather similar. That is, participants create only one hypothesis based on the evidence up to that point, and add extra links to accommodate new or contrary information. However, these additional links are only local solutions and the complete hypothesis is not revised globally, as the current hypothesis fits the data (although it contains spurious edges).

The similarity between the two learning processes, with the different cues available, can be pushed further. The idea behind the unification is that temporal order information restricted to two consecutive time units results in the same inference for the participants as interventional data.[50] More precisely, if at time t a node X is active while at $t+1$ another node Y is active, then participants consider X to be causally effective and infer a link $X \to Y$, just as in the case when they see an intervention on X activating Y. Reading temporal order this way and using the heuristic model of Fernbach and Sloman is in accord with the explanation given above by Lagnado and Sloman and predicts the patterns on Figure 11 rather closely. More precisely patterns 2 and 4 exactly, pattern 3 with one additional link, and it is far off on pattern 1.

Consider first, for example, patterns 2 and 4 in Figure 11. In case of pattern 2, when all computers get affected, a participant will first observe consecutive pairs in the order of (A, B), (B, D) and (D, C) where the first letter in the parentheses occurs first. If it is treated as data from separate interventions where the first letter is intervened on and activates the second one in the parentheses, the heuristic model predicts the same $A \to B \to D \to C$ chain as Lagnado and Sloman hypothesizes. Let us assume that the participant then observes a temporal order of $(A, B, C, \sim D)$, which is nothing but the observation of

[50] Even those cases will be handled where more time passes between the events. See the next footnote and the rules below.

(A, B) and (B, C) after each other.[51] If we, again, treat these as interventions while the participant has the chain as their current hypothesis, based on the heuristic model, the (B, C) "intervention" is not yet accounted for and predicts that participants will endorse a further $B \to C$ link, which completes pattern 2. This leads to a confound structure (without revision), similarly as in the case of the heuristic model, and accounts for the experimental data.[52] In case of pattern 4 the consecutive pairs of temporal order the participants observe are (A, B) and $(B, C \textit{ and/or } D)$. If, again, these are treated as interventions (where $(B, C \textit{ and } D)$ is in intervention on B which activates both C and D), then the heuristic model predicts a link from A to B and a common cause structure on B, and C and D. Exactly like in pattern 4.

Thus, indeed, this unified model predicts patterns 2 and 4 exactly. In case of pattern 3, it predicts this pattern with a link from A to C from observations of the (A, C, B) temporal order instead of a (rarely chosen) link from B to C.[53] However, pattern 1 does not line up well with the prediction of the generalized heuristic model, which predicts a common cause structure with A as the source. The original heuristic model is restricted to three nodes and very few observations. This is the only scenario in these studies when four nodes can be active at the same time. It is not clear whether it shows that humans apply more complex heuristics for larger structures; are removing edges after a large number of observations; or their hypotheses were effected by something else, by the cover story for example.

Let us now turn to the implementation of these models in terms of the computational dynamical systems framework. Again, just as in the case of parallel computers, there is no need for secondary causal neighborhoods and determined regions to accommodate these heuristic models. Throughout the literature it is always assumed that all the possible causes and variables are known from the beginning. That is, there is no need for the expansion of the models via new atoms. The presentation will be on the level of graphs and operations thereon, as the encoding of graphical models in terms of hereditarily finite sets were shown above.

Throughout the implementations it will be assumed, just as in case of the experiments, that the participants can distinguish whether events happened due to

[51] It is not entirely clear from the description of the experiments whether the observed temporal order is $(A, B, C, \sim D)$ as quoted by Lagnado and Sloman above, or possibly $(A, B, \sim D, C)$. The possible difference is that in the latter case there could be a longer break between B and C, namely the observations in the following seconds could be simply (A, B), (B, *no change*) and (*no change*, C). The rules introduced below will accommodate such effects (and also deal with the so called "launch effect"), but here I will use the most simple description to make it easier to convey the main idea behind the unification.

[52] It has to be remarked that here the confound structure is reached by first inferring a chain and then the addition of a spurious edge, while in case of interventions it was first a common cause structure later supplemented with an extra edge. However, the explanation is the same, i.e. one current hypothesis, only addition of edges without revision, and the temporal order information makes the original chain hypothesis plausible.

[53] It is surprising that that link is not endorsed by the participants as that is the only way to account for such observation, and through 100 test they are very likely to witness such a scenario.

an intervention. The heuristic causal learning will be implemented in two steps. First, based on the observations an internal representation will be constructed. This step is essentially the unification of the interventional and temporal cues as indicated above. Second, the heuristic rule (or rules) of Fernbach and Sloman will be applied to the representation.

The first step in the process is purely operational. That is, it does not have any psychological fidelity and it is not driven by such considerations at all. Its sole purpose is to lead to a unified representation of observations in both cases of causal cues. The second step does have psychological fidelity to the extent Fernbach and Sloman's heuristic model has, as it directly reproduces their heuristic rule(s). Thus the main aim of the following rules is to provide a computational model in the framework of computational dynamical systems that mimicks the behavior of the participants and reproduces the data from the experiments, but does not claim psychological fidelity.

Now we turn to the introduction of such production rules. First those rules will be introduced that reproduce Fernbach and Sloman's heuristic model (that is, the second step above). Thus now it will be assumed that based on their observations there is a representation available for the participants, where interventional and temporal order cues are unified.

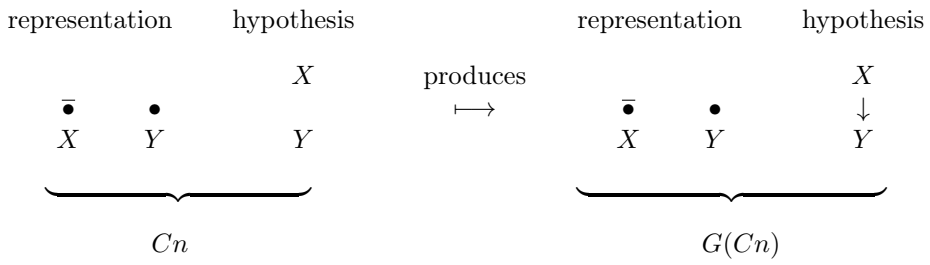
Participants will have separate fields for their observations, working memory (where needed) and their representations. Beyond these fields they will always have *one* current hypothesis about the causal structure that will be locally updated. The variables that stand for the events will have four different states: (i) inactive; (ii) active; (iii) activated by an intervention; (iv) causally effective. The fourth one here is the unified notation for signaling that a variable was either intervened on, or became active during the previous time unit and is now causally relevant. (iii) will be present only in observations, (iv) will be present in the unified representation and in the working memory only. These states will be represented in the following way.

- $\circ$ inactive or "off"
- $\bullet$ active or "on"
- $\check{\bullet}$ active or "on" by being intervened on
- $\overline{\bullet}$ causally effective

The current hypothesis of the participant will be represented as a graph, containing one node for each variable.

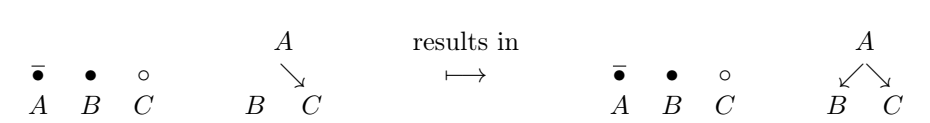
The main rule of the heuristic model was described above as follows. A causal link is added between the causally effective variable and another variable, if the other variable is represented as a regular active one.[54] As the hypothesis is the union of all causal links implied during the trials, it means that once a causal link is added to the hypothesis, it will not be removed later. This main rule can be captured by the production rule below:

[54] All in the experiments that are discussed here there is only one causally effective variable at a time, hence the use of "the causally effective variable." However, nothing excludes the possibility of multiple simultaneous causally effective variables. See the end of this section for comments and further directions on this issue.



As a reminder, the number of links that can be added after an observation is not restricted in this model. This aspect is also captured by this production rule. If, say, an intervention on X activates both Y and Z, then both pairs, i.e. (X and Y) and (X and Z), fit the causal neighborhood of the rule. As a consequence, both links, i.e $X \to Y$ and $X \to Z$, will be added in parallel.

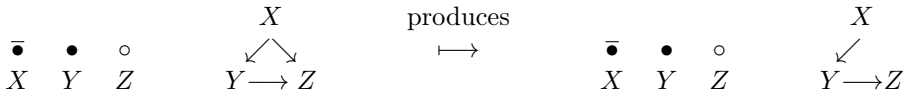
Let us see how this rule works in an example. Let us assume that a participant, whose current hypothesis is a one link model $A \to C$, observes an intervention on A which activates B but not C. Thus A gets represented as causally effective, hence by $\overline{\bullet}$. Based on the heuristic model, to accommodate the new information, a new $A \to B$ edge has to be included in the hypothesis graph. As the nodes A and B fit the causal neighborhood Cn of the rule above, it indeed results in the required new hypothesis:



This one rule reproduces the heuristic model that fit rather well, better than Bayesian models, the experimental data from Experiments 1 and 2. It reproduces the bias in human learning that leads to confound structure on data generated by chains and never leads to a mistake on a common cause structure. This one rule also reproduces most of the experimental findings of Lagnado and Sloman.

However, in Experiment 3 of Fernbach and Sloman participants were trained to modify their hypothesis in the face of "unlikely" evidence, i.e. when they saw a "diagnostic intervention." Some participants did also apply that refinement of their hypothesis without training in Experiment 2. Hence the following rule recreates the (possible) impact of the training in case of "diagnostic interventions." That is when a participant has a confound structure as a part of their hypothesis, but observes something that is way more likely assuming the data was generated by a chain and not a confound structure. In that case the agent (sometimes, see above) removes the spurious edge:

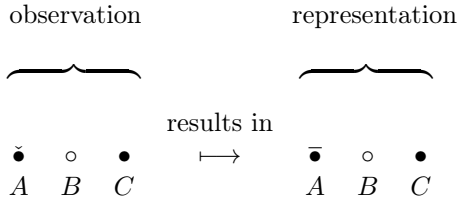
representation hypothesis representation hypothesis



This rule also reproduces the time sensitivity effect of when then diagnostic intervention is seen in the second experiment of Fernbach and Sloman. Namely that participants in the "Early Group" inferred confound models most of time and it was only the "Late Group" that was impacted by observing a diagnostic intervention. Fernbach and Sloman hypothesized that the reason is that in the case of the "Early Group" there was no current hypothesis that was confronted by the diagnostic intervention, while in the case of the "Late Group" participants already assumed a confound model. This rule reproduces that effect, as it is only applicable if the current hypothesis already contains a confound structure as a substructure. In case of an empty hypothesis, the case in the "Early Group" in the experiment, the first rule applies, leading to a one-link structure as the new hypothesis.

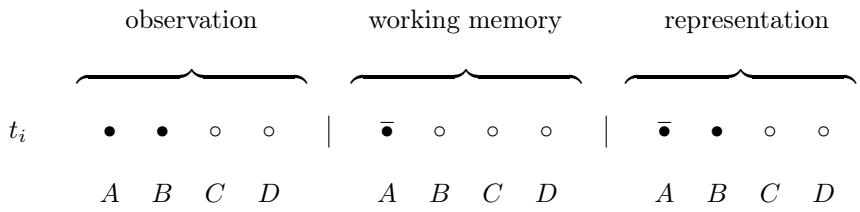
Again, this latter rule is not applied by everyone, not even after the training. It means that to reproduce experimental data, the application of this rule should be randomized. Where randomization can be done for each participant by a semi-random generator as it was indicated above. This concludes the rules that reproduce the heuristic model. Now we turn to the unified representation of the observations of the participants in the different cases.

In the experiments of Fernbach and Sloman the participants observed a system where there was no time delay between the intervened on variables and their effects. Furthermore, they did not interact with the system, merely observed it. This means that the representation on which the heuristic rules will act can be constructed based on one observation. That is, the variables that are inactive or become active but not as a result of being intervened on simply have to be copied into the representation. The intervened on variables have to be encoded as "causally effective" in the representation. Thus, for each variable X, its representation only depends on the state of X in the observation, that is, the representation can be constructed for each variable locally, in parallel. As the representational rules for temporal cues will be more involved, these rules will be merely indicated by an example:



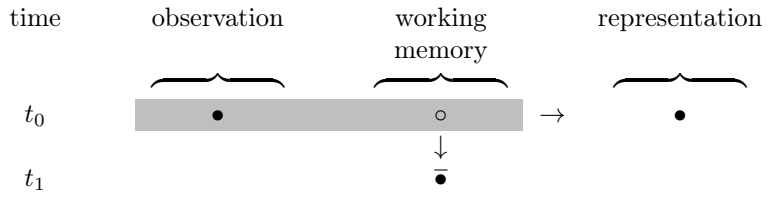
Let us now turn to Lagnado and Sloman's data on causal structure learning when temporal order information is available. The main idea behind the

unification of this model with the previous one is to treat the information from consecutive time moments in the same way as interventions were. That is, the task here is to convert temporal information data to the same format as in case of the interventional observations. At each time the participant has to remember which variable became active in the previous second, and observe which variables became active in the current one. Then combine this information to add edges from the previously activated node to the current ones. To achieve this, participants will have three fields besides their current hypothesis. An "observational" field which they cannot influence. A field of "working memory" where they remember which node or nodes became active in the previous second. Thus working memory encodes only important information from the previous state and not purely a "copies" it. Then the observation and the content of the working memory is combined into a representation. It is a representation on which the main rule of the previous heuristic model can operate. At any time t_i a participant's observational field and working memory will be represented in the following way:



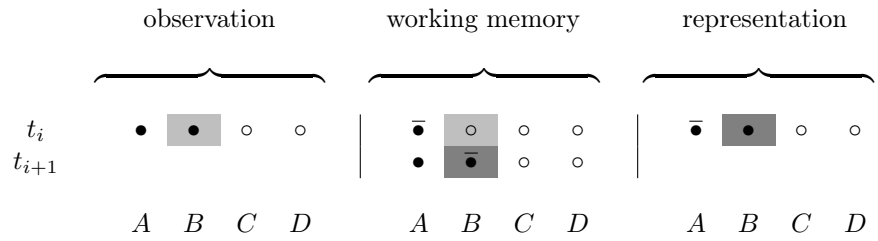
Thus, at time t_i variables A and B are active as shown by $\bullet$. The participant remembers (working memory), that in the previous second, i.e. at t_{i-1}, A did become active, which is represented as being causally effective at t_i, indicated by $\overline{\bullet}$. So the participant creates the representation in which A is marked as a causally effective variable which activated B, hence, by the rule of Fernbach and Sloman's heuristic model, an edge $A \to B$ is added to the current hypothesis, if it is not yet included. We now turn to the rules that will result in such a representation.

At time t_i the participant has an observation and remembers the important information from t_{i-1}. Now they have to do two things based on the information available. One, to create the representation, so the structure learning operations can be applied. Two, they have to remember the important information for the next second, t_{i+1}. In the example above B just became active, so for time t_{i+1} they have to remember it as the new information and represent it as $\overline{\bullet}$. What is important about these operations is that they always depend only on one variable at a time. More precisely, the state of variable B in the representation and in the working memory at t_{i+1} depends only on the state of variable B in the observational field and in working memory at time t_i. The rule below captures the operation that was just described in case of variable B:



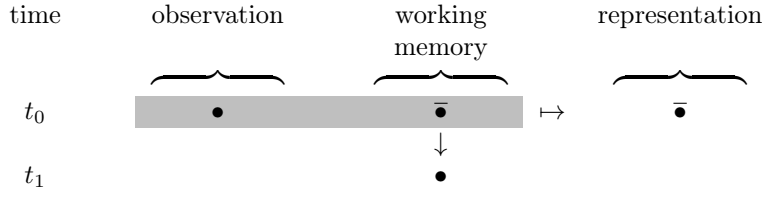
The information available at t_i, i.e. the causal neighborhood Cn of this production rule, is indicated in the gray rectangle above. That is, the variable in question was inactive at time t_{i-1} as it is indicated in the working memory by $\circ$, and it just did become active as it is a $\bullet$ now in the observational field. Hence, at the current time it has to be shown that it is active (which means that it became activated by a "causally effective" variable) in the representation, and for the working memory at t_{i+1} it has to be encoded that it just became active and thus has to be treated as an "causally effective" variable, as indicated by $\overline{\bullet}$.

The application of the above rule has the following impact in the above case. Again, the causal neighborhood is shown in gray, while the result of the operation, the determined region is indicated by a darker gray:



Rules will apply for each variable at the given time and will be executed in parallel.

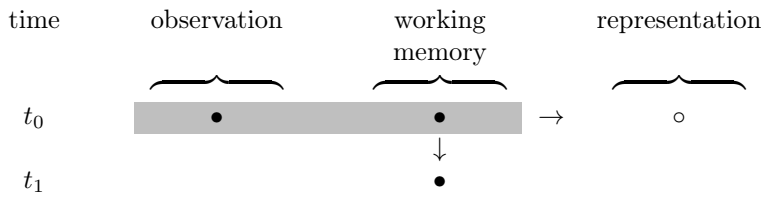
There are actually only a few cases, all of which can be treated by very similar rules. The other interesting case is to "remember" to deal with variables that were causally effective in the previous second. That is, what happens to variable A in the example above and what will happen with B in the next second:[55]



[55]This rule also applies in the beginning, when participants actually intervene on computer A, and could handle further cases, would the participants observe or create further interventions.

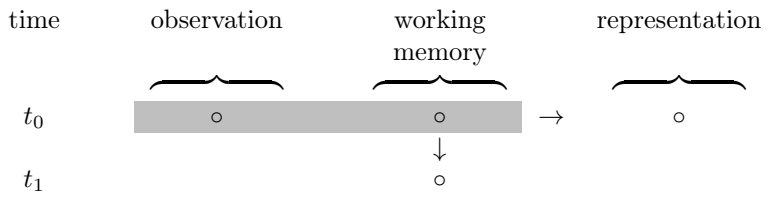
Variable A is currently active, but it became active in the previous second thus it has to be treated as a causally effective variable. For the working memory it only has to be remembered that it was on at t_i but it did not became active earlier, and will have no further causal relevance.

Once a variable became activated more than one second ago it is not of interest anymore, as it will be causally irrelevant. This situation is handled by the following rule:



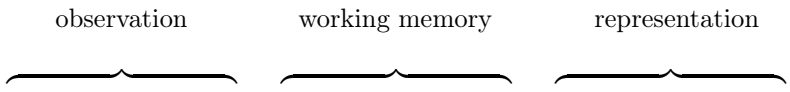
The information that it is still active is passed on to the next second, but now it is turned into a $\circ$ in the representation, as it is not interesting anymore for the causal structure.[56]

The only remaining rule handles those inactive variables that stay inactive:

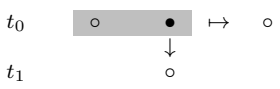


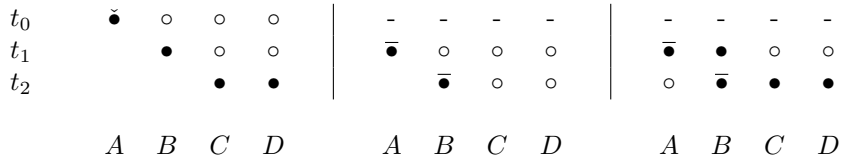
This rule merely keeps track of the inactive variables and is there to "complete" all fields of information.

Let us now turn to a full example which recreates pattern 4 in Figure 11, i.e. where the computers are activated in the order of A, B, and C and D simultaneously.



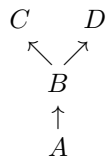
[56] In the experiment once a computer became active, i.e. crashed by a virus, it did stay activated. The case where variables can become inactive after some time can also be handled with the following rule:





At t_0 the fields of working memory and representation are empty. For t_1 the content of the observation from t_0 is merely copied into the working memory at t_1 in case of $\bullet$ and $\circ$, while interventions are represented as $\overline{\bullet}$.[57] The empty places should contain an $\bullet$, but those were omitted this case to take the relevant information more transparent. The hypothesis for the graphical structure at the beginning is the empty graph on four nodes, which is not shown above.

At t_1 the participant remembers that variable A was intervened on at the previous second and is thus causally effective, and now observes that B just became active. Hence A and B fit the causal neighborhood of Fernbach and Sloman's rule and results in the addition of the $A \to B$ edge in the hypothesis graph. At t_2 the participant remembers that in the previous second variable B became active and now treats it as the causally effective variable. At t_2 both C and D became active, which based on the heuristic model indicates a common cause structure $C \leftarrow B \rightarrow D$. The addition of these two edges to the current hypothesis results in the graph:



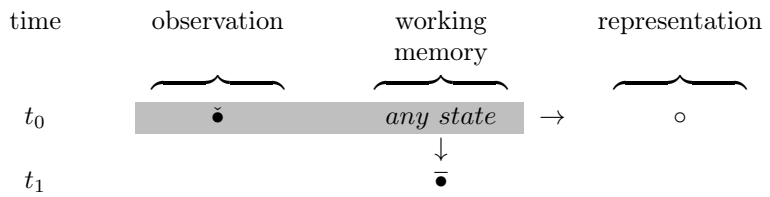
Which matches pattern 4 of Figure 11 exactly.

These rules are sufficient to reproduce the findings of Fernbach, Lagnado, and Sloman in a narrow way. In the following a few further rules will be given to make this computational model somewhat more complex and even more realistic. Two separate issues will be handled by these new rules: (i) interventions during but not at the beginning of an experiment, and (ii) temporal delay.

Both in Fernbach and Sloman's and Lagnado and Sloman's experiments interventions only happened at the beginning. That is, in Fernbach and Sloman's experiments the events were simultaneous, but the interventions were assumed to be the first events under the implicit temporal order. In Lagnado and Sloman's experiment participants always intervened on computer A at the beginning of each trial. However, in many other experiments participants are allowed to intervene during a series of events. The next rule handles this situation.

[57] These rules are omitted as they are rather straightforward based on the previous ones. In the experiment every trial starts with an intervention. If participants would observe a working system without an intervention in the beginning, then to deal with the starting case, rather similar rules than the ones above have to be introduced, where the field of working memory is empty.

As before, it is assumed that agents are capable of distinguishing an intervention from a mere activation. When an intervention is made, the previous history of that event becomes irrelevant. This is indicated by "*any state*" in the rule, that is, the content of the working memory does not play a role in the outcome of the rule.[58] The fact that the variable became active by an intervention should not have consequences for the causal hypothesis of the agent at the current second. For, it did not become active due to the causal impact of another variable. This is captured by encoding the variable as ∘ in the representation, thus no heuristic rule can be applied to it at t_0. Finally, the intervened on variable has to be considered as casually effective from the next second. This features are captured by the following rule:



Thus this simple rule handles interventions that happen during, but not at the beginning of a series of events.

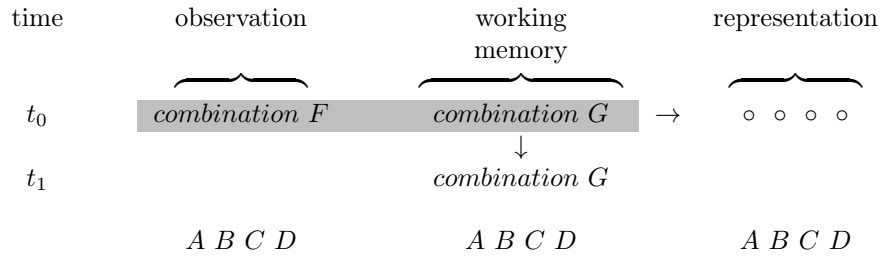
One aspect of interventions that has not been addressed is how agents choose interventions when they are allowed to. The experimental data from (Lagnado and Sloman 2006) and (Steyvers, Tenenbaum, Wagenmakers and Blum 2003) indicated that people prefer source nodes over sinks on a general, heuristic level. These interventions are not always the most informative ones, which also shows a confirmation bias towards positive hypothesis testing, as opposed to disconfirmation. The ways in which agents choose interventions was further studied in recent studies, e.g. Coenen, Rehder and Gureckis' (2015) and Bramley, Lagnado and Speekenbrink's (2014). These studies provide computationally specified Bayesian models to reproduce experimental data. Thus, again, as in the case of computationally specified causal structure learning models, e.g. Spirtes et al. (1993), these models will not be discussed as the previous section on (parallel) computing reasonably covers those cases. After dealing with interventions in general, now we turn to temporal delay.

The experiments of Lagnado and Sloman about the computer network were treated above as if the events were evenly distributed in time, as the authors were somewhat ambiguous in those cases when certain variables were not activated. For example, in case of pattern 2 in Figure 11 it is not entirely clear from their description of the experiment whether the observed temporal order was $(A, B, C, \sim D)$ as quoted by Lagnado and Sloman above, or $(A, B, \sim D, C)$. The possible difference is that in the latter case there could be a longer "break," a temporal delay between B and C, namely the observations in the following seconds could be simply (A, B), (B, *no change*) and (*no change*, C). This difference is important as we do know that temporal delay between two events

[58] Technically it means that this is a rule scheme, and it covers three rules at the same time, as a variable can have three different states in the working memory.

impacts our assumptions about causal structure. For example, if a light turns on almost immediately after turning a switch, most people infer that the switch turns on the lights. If, instead, there is a five second delay between the two events, the causal relationship is usually not assumed. Cleary, the production rules above are not set up to handle such effects.

The rule scheme below deals with a one unit "skip" or "delay." The main idea is to ignore that second when nothing new is happening in the observational field. It can be formulated in the following way: there is a current combination F of states of variables in the observational field and a combination G in the working memory. Where F and G are either identical, or those variables that are encoded as causally effective in the working memory, i.e. as $\bar{\bullet}$, are all just regular active nodes in the observational field. Thus the diagram below indeed covers several production rules at the same time:



Indeed, the result of this rule is that when the agent recognizes that nothing new happened in a particular second, they just keep everything in their working memory, but "empty" their representation, so no heuristic rule can apply at this point. It has to be pointed out that the causal neighborhood Cn in gray in the picture clearly contains the causal neighborhoods of the previous rules. As a consequence, based on the convention that the rules with the largest inclusive causal neighborhood are applied,[59] there cannot be a situation where both this rule and some of the ones above with overlapping causal neighborhoods are applicable at the same time.

This rule not only ignores a one second time delay, actually it ignores every time delay, irrespectively of its length. To accommodate the impact of the length of the time delay, a clock or counter could be included in the model once the observational field is "empty." Then, when the observational field is "empty" and there is a counter, it gets raised by one, i.e. it counts the seconds that passed since nothing new happened. Then, if the counter reaches a pre-set number, i.e. long enough time passed that people consider the previous events causally ineffective, the working memory should be "erased" and the counter be removed. Or if something "happens" in the observational field before the counter reaches the pre-set value, then the counter gets removed and the previous rules apply. As these actions can be achieved by similar rules as the ones above and those involving a clock in the previous section on parallel computing with a little ingenuity, they will be omitted here.

[59] This convention was introduced in the previous section on parallel computing, following Sieg's (2009).

With these additional rules the description of this heuristic model as a computable dynamical system is finished. However, with the aid of this computable model further questions can be raised and experiments planned to investigate them. We now turn towards those questions.

The genericity of the production rules above, namely that they are applicable on graphs of any size, helps to design further experiments, to test their validity further, and to gain more information about human causal structure learning. First of all, lets go back to pattern 1 in Figure 11 of Lagnado and Sloman, the only pattern where the prediction of the unified model is way off. This is the experiment where all nodes become activated simultaneously. At the end participants learned the correct causal structure while the heuristic model predicted a common cause structure with A as the source. That is, it seems that participants recognized that the $A \to C$ and $A \to D$ edges are not necessary to explain the observations. As a comparison, in the experiment of Lagnado and Sloman they observed 100 interventions, which is rather high compared to the 5 interventions showed by Fernbach and Sloman. This suggests that participants do recognize and utilize that they never see trials in which only (A, C, D) or (A, C) or (A, D) are active in themselves, which is extremely unlikely under a common cause structure through 100 trials. Thus, it seems that participants used "negative" or lack of information in this case, contrary to the reasoning behind the heuristic model. It should be tested whether participants do ever have the common cause structure as hypothesis and remove edges later or not. That is, participants should regularly be asked about their current hypothesis during the 100 trials. Another possible idea would be that participants simply ignore all those cases where all of the nodes become activated simultaneously, because it does not convey any information.

Another question that is not addressed by experiments is that what happens when there is more than one cause that is active at the same time. In each experimental situation it is at most one variable that is intervened on, and two or more variables became active only in the last round, so they do not have further consequences that have to be separated. It would be interesting to see whether participants can learn sparse structures on four variables where there is more than one cause. For example in case of a simple two edge structure where $A \to B$ and $C \to D$, the heuristic model predicts $B \leftarrow A \to D$ and $B \leftarrow C \to D$, i.e. two common cause structures at the same time. It would be interesting to see how effective human learners can be on slightly bigger graphs in these more complicated situations, how they choose interventions if they are allowed to and so on. It seems plausible that the very simplistic heuristic model will be incorrect in some of those cases, which could lead to the improvement of heuristics or the discovery of further ones.

5.4 Exploring the Borders

In this section cognitive models of human cognition were examined. I focused on heuristic models, as the previous section showed that realistic models of parallel computing are Computable Dynamical Systems, indicating that the remaining

cognitive models do fall under that notion as well. However, the computational description of heuristic causal structure learning turned out to be simpler than was expected. In this subsection my aim is to give directions for further work and provide a broader picture of models of human cognition and human processes in general that most likely would require more complex Computable Dynamical Systems to capture, or might be beyond their scope altogether. First I describe a further direction, analogical learning and reasoning (which can be connected to causal learning as well). Then I turn to models of category learning and feature representation, which most likely would require the use of secondary causal neighborhoods and determined regions when represented as Computable Dynamical Systems. Finally, I will look at theories of emotion, which might be outside the scope of Computable Dynamical Systems.

A Further Direction: Analogical Learning and Reasoning

An analogy is a structural or relational similarity between two sets of objects. Analogical reasoning is used to suggest inferences. This kind of reasoning is ubiquitous in human reasoning and cognition. The cognitive models of analogical reasoning have two main components: (a) structural alignment and (b) analogical inference or inference projection. The aim of the structural alignment is to find a shared relational structure between the base and target. Thus, it is important that the corresponding objects in the two representations are in matching systems of relations. The emphasis here is on the structural resemblance of the systems, not actual resemblance of objects. This alignment is constrained in two ways. By the structural consistency requirement, which requires a bijection between the aligned substructures of base and target. And by the systematicity principle, which prefers correspondences where not only the lower level objects match each other but their relational structures as well. The next step is analogical inference. Candidate inferences are sought by structural pattern recognition: "once the base and target have been aligned and their common structure found, if there are additional assertions connected to that common structure in the base (and not yet present in the target), then this structure will be brought over as a candidate inference." Of course the search for inference candidates is highly selective, being constrained by semantical considerations for example. (Gentner and Smith 2013)

In their (2015), Goldwater and Gentner were interested in the question "how people come to be sensitive to abstract causal patterns." They found that analogical comparison leads to an increase in detecting causal patterns across diverse examples. That is, people who were trained by structural alignments of such diverse examples became more attentive to causal patterns than those without training. These findings could possibly deepen the understanding of causal structure learning of humans, as it seems that abstract or analogical knowledge of causal patterns contributes to whether observations are represented in terms of causal relations. At the same time, of course, this abstract knowledge is influenced by observations (or by more direct training). Thus, it seems that these findings lead to a more complex picture, where different factors influence

each other simultaneously, i.e. in parallel. More precisely, observations have an impact on their representations as well as on one's abstract causal knowledge. At the same time, this abstract knowledge seems to contribute to how the observations end up being represented.

My aim is to look further into causal structure learning through analogical comparisons, and into analogical reasoning itself. The inclusion of analogical learning and reasoning would most likely also lead to more involved Computable Dynamical Systems, with more essential parallelism, and possibly even with secondary causal neighborhoods and secondary regions. For more on the use of secondary structures, in another context, see the next subsection. A direction that could be promising is to use mathematical experience of structural characterizations, such as Dedekind's structural axiomatization of simply infinite systems, axiomatizations of algebraic structures such as groups and fields, or as Sieg's structural axiomatization of the concept of computability to deepen our understanding of analogical learning and reasoning.

Secondary Structures

When I discussed parallel computers and the cognitive models of causal structure learning, I emphasized that both do not need secondary causal neighborhoods and determined regions, as their structures do not expand over time. We saw in the previous paragraph on analogical reasoning that in case of looking for inference candidates there might be a need for secondary structures. Now I mention some cognitive models which most likely would have to exploit the secondary structures of Computable Dynamical Systems. These models come from category learning and feature representation.

In their (2004), Love, Medin and Gureckis proposed a model of category learning employing flexible structure search. Their model, SUSTAIN (Supervised and Unsupervised STratified Adaptive Incremental Network), learns categories from examples. SUSTAIN starts out with a simple structure, assuming only one cluster and recruits further clusters in the face of surprising information. That is, SUSTAIN introduces a new cluster either when it receives feedback that a wrong categorization was made (supervised learning), or when presented with an example which does not fit any of the existing clusters closely enough (unsupervised learning).[60] As this model of category learning is flexible and does not have initial bounds on the number of categories that will be learned, and, more importantly, no bounds on the number of clusters that can or have to be used for the categorization, SUSTAIN is an example of a computable model of cognition that would have to exploit secondary causal neighborhoods and determined regions when described as a Computable Dynamical System. This is due to the fact that every time a new cluster is introduced, at least one

[60]It is possible that a category is composed of multiple clusters; it happens in cases where the category can be exemplified by rather different objects. For example the category 'spoon' has as its elements small metal spoons as well as large wooden spoons. These rather different kinds of spoons might end up being in two separate clusters while still belonging to the same category. Thus the number of clusters can grow even without new categories being introduced.

new atom has to be introduced to represent that cluster, and its place in the structure strongly depend on the previous structure.

The other example of a cognitive model that would most likely have to use secondary structure when described as a Computable Dynamical System is that of flexible feature representation. To capture the flexibility of human cognition, researchers seek models where the set of possible features can adapt with new experience, instead of having a pre-defined set of them to represent all possible objects (Austerweil and Griffiths 2013, p. 818). These models have a stochastic process called the Indian Buffet Process (Griffiths and Ghahramani 2005 and 2011) at their core. Fortunately, to understand the feature of the process that would most likely require secondary structures does not require a complete technical description of it. Flexible feature learning is formulated in terms of a matrix factorization problem. To accommodate the unbounded number of features from finitely many observations of finitely many objects, these processes use a matrix with a finite number of rows (the objects) and an "infinite" number of columns (the features), where at any point, only finitely many columns contain anything besides 0s. Of course, only those columns that contain non-zero elements impact the computation. Such matrices are constructed by the Indian Buffet Process, which is described as follows:

> N customers enter a restaurant one after another. Each customer encounters a buffet consisting of infinitely many dishes arranged in a line. The first customer starts at the left of the buffet and takes a serving from each dish, stopping after a $Poisson(\alpha)$ number of dishes as his plate becomes overburdened. The ith customer moves along the buffet, sampling dishes in proportion to their popularity, serving himself with probability $\frac{m_k}{i}$, where m_k is the number of previous customers who have sampled a dish. Having reached the end of all previous sampled dishes, the ith customer then tries a $Poisson(\frac{\alpha}{i})$ number of new dishes. (Griffiths and Ghahramani 2005, p. 5)

Thus, each "customer" samples some new "dishes" that were not sampled by anyone else before. To handle the new "dishes", i.e. the new features, a new atom would have to be introduced when described in terms of Computable Dynamical Systems. As the emphasis is on the newness of these features and there is always only finitely many of them "in use", they can clearly be accommodated with secondary causal neighborhoods and determined regions.

These cognitive models of category learning and feature representation are clearly computable. The reason for mentioning them here was to exhibit some cognitive models that most likely would have to use secondary causal neighborhoods and determined regions when described as Computable Dynamical Systems. These models seem to answer Sieg's question: "Are there mental processes for whose representation this aspect [i.e. secondary neighborhoods and operations] of Gandy machines might be crucial?" affirmatively. This makes the question whether there is a natural subclass of Turing computable functions in which these Computable Dynamical Systems lie even more significant.

Possibly Non-computable Theories

In order to provide a rather broad picture, the aim of this subsection is to look at human processes that might not be possible to be put into a computable framework. More precisely, I take a quick look at theories of emotion which might be non-computable or contain a non-computable component.

Theories of emotions are good candidates for possibly non-computable theories because even cognitivist theories of emotions usually include a non-information-processing or non-cognitivist component with varying importance (Oatley and Johnson-Laird 2013). Furthermore, there are non-cognitivist theories of emotion as well, providing neuropsychological and physiological explanations. These components might turn out not to be computable.

A cognitive theory of emotions that relies strongly on non-cognitive explanation is the so-called core-affect theory. According to this theory, underlying emotions is a neurophysiological state called the core affect. The core affect is characterized along two continuous dimensions: (1) from pleasure to displeasure; (2) from high arousal to low arousal (Oatley and Johnson-Laird 2013). Examples of non-cognitivist, somatic theories of emotions include Damasio's (1994) and Prinz's (2004). According to Damasio, feelings and emotions are mental experiences of body states. These so called "somatic markers" then in turn influence decision-making and behavior. These somatic markers can occur on the neurological or on the physiological level. Prinz similarly states that emotions are perceptions of body states, embodied appraisals, and emphasizes that "bodily changes precede our emotional experiences." (2004, p. 4) Prinz argues that most of the time emotions are not cognitive, that "they are not generated by acts of cognition, and they are not conceptual." (p. 50)

As we see, all these theories take changes of the body into account. Of course, nothing guarantees or necessitates that these functions and changes of the body can be described and represented by computable functions. Thus, these theories provide examples of theories which might turn out not to be computable. For example, representing core effects with an open or closed unit circle, containing uncountably many points of $\mathbb{R}^2$ would clearly violate the condition of finiteness. However, as there is no detailed, formal description available of these theories, it is not clear whether these components can be described or simulated by discrete means, just as in the case of analogue machines.

At the same time it has to be remarked that there are several computational models of emotion, as surveyed in (Marsella, Gratch, and Petta 2010). Although most computational models are based on cognitivist appraisal theories, even Damasio's theory motivated a computational model, namely Velásquez's *Cathexis* model (1997). However, it has to be noted that though Cathexis takes affects and arousal into account, it has little impact on the behavior of the system, which is mostly governed by Frijda's (1986) cognitive theory of action-readiness. Furthermore, the changes and feedbacks of affects and arousal are left mostly unaddressed.

6 Further Directions

In this short section I would like to point to some further directions, some of which I intend to take up in the future. I mention three, I believe interesting, directions, two concerning theoretical models of computation and one concerning models of cognition.

It seems that the formal description of analogue (non-quantum) computing devices is rather lacking. The description of such devices, beyond the mere statement of which kind of differential equation they are solving etc., would help to address the question of their simulation. That is, we saw in *Exploring the the Borders* of Section 4 that while most scholars believe that analogue computers can be routinely simulated by digital ones, the analogue models are criticized for their exploration of infinite precision. I believe that precise models of analogue computational devices would contribute to solving such issues.

In the future I would like to continue working on the model of parallel computing as a Computable Dynamical System. I see two interesting directions for possible continuation of Section 4. One, providing a more formal description of the computational tasks at hand which could lead to formally proving that the particular Computable Dynamical System indeed realizes them. Two, on a somewhat informal level, addressing features of programming languages in this framework. For example, the "chain of natural numbers" was used in a way that resembled a for loop. Extending this direction might lead to interesting results.

In the area of models of cognition I became interested in the models of analogical reasoning while learning about them. Wilfried Sieg pointed out that it seems that these theories, establishing analogies in terms of isomorphisms between bases and targets and abstracting from them, could be improved by building on mathematical experience. That is, the mathematical experience of recognizing similarities (analogies) between mathematical structures and proofs and using them to introduce abstract concepts, such as groups, capturing their common features. In the future I intend to undertake this direction jointly with him and David Danks.

Appendix: Kreisel on Incompleteness and Church's Thesis

According to Kreisel, foundations of mathematics should provide a theory of mathematical practice and the mathematical experience of the working mathematician. Hence, Gödel's interpretation of the incompleteness results bearing consequences for the mathematical capabilities of humans and machines is rather similar to Kreisel's approach to mathematical experience. Kreisel thought that transfinite progressions or some other mathematical theories might provide a better theory of our mathematical experience. At the same time Kreisel understood Church's Thesis as stating a limitation of human capabilities. He was interested whether such an interpretation of the Thesis can be refuted or justified based on our knowledge of transfinite progressions, assuming that these progressions do provide a theory of mathematical experience and practice.

This Appendix provides a critical analysis of Kreisel's general philosophical approach to mathematics, his interpretation of transfinite progressions, and their relevance for his understanding of Church's Thesis. Kreisel's work was examined in detail because he addresses several issues that are central to the Dissertation. That is, he focused on human cognition restricted to mathematical practice investigated through the notions of human and mechanical effectivity and its relevance of Church's Thesis. However, Kreisel's approach is more narrow than that of the Dissertation; it is essentially restricted to interpreting results from mathematical logic. Furthermore, as will be clear from the following critical overview, Kreisel did not provide a convincing analysis even by his own requirements. As a consequence, though considerable in length, this section is only included as an Appendix and not as part of the main text.

Kreisel's General Philosophical Views[61]

Mathematical logic has two aspects for Kreisel. First, it is a part of pure mathematics. The second aspect, its role for the philosophy of mathematics, is compared to the role of ordinary mathematics for "what used to be called natural philosophy."[62] Natural philosophy analyzes our experience of the physical world, whereas philosophy of mathematics analyzes our mathematical experience. As Kreisel says, "perhaps primarily, it is intended as a tool in the philosophy of mathematics"! (1967, p. 201) The role of the resulting foundational studies is described as follows:

> Foundations provide a theory of mathematical practice. Thus, in the first place, they describe and analyze our mathematical experience,

[61] The following overview of Kreisel's general views is based on his (1967), (1967b), and (1965).

[62] Sometimes Kreisel uses 'natural science' instead of 'natural philosophy'. In the following I will use 'natural philosophy' everywhere, as it seems to be more accurate. Since "what was used to be called natural philosophy" would also include 'philosophy of (natural) science' than just 'natural science'.

> that is the body of results and methods as they present themselves to the working mathematician; as any other theory, foundational results lead to an extension of mathematical practice. (1969, p. 5)

To have a better understanding of what is meant by a theory of mathematical practice, let us take a look at Kreisel's more detailed description of what mathematical experience consists of, i.e. the mathematical experience of the working mathematician:

> The data of foundations consist of the mathematical experience of the working mathematician. This presents itself as being about certain mathematical *objects* such as natural numbers or groups; the processes involved in mathematical activity present themselves as primarily *mental* and *abstract* (not sensory), but they are recorded and communicated by use of *concrete*, that is, spatiotemporal *symbols*. (1965, p. 95)

It should be emphasized here that "the processes involved in mathematical activity", such as reasoning, are considered to be "primarily *mental* and *abstract*". We will return to the question of mathematical reasoning in detail later, as it will be important for Church's Thesis.

How should experience, physical or mathematical, be approached? According to Kreisel, in the case of both natural philosophy and philosophy of mathematics the process of dealing with a particular problem of experience (ideally) consists of (i) figuring out "what formal problem constitutes the correct formulation" of it and then (ii) obtaining a solution of the formal problem. (1967, p. 204) Progress on the first task can be made by what he calls *informal rigour*, that is, "[t]he 'old fashioned' idea is that one obtains rules and definitions by analyzing intuitive notions and putting down their properties." (1967b, p. 138) According to him "[t]his is exactly what mathematicians thought they were doing when defining length or area or, for that matter, logicians when finding rules of inference or axioms (properties) of mathematical structures" (p. 138). Other examples of such analyses mentioned by him are the definition of 'dimension' in topology to capture the intuitive notion of dimension, and the concept of recursive function to capture mechanical processes. According to Kreisel:

> [T]he only *obvious* purpose of introducing the concepts mentioned in a formally rigorous way is *as* [emphasis by me] correct[63] analysis of the given intuitive concept![64] (1967, p. 205)

At the end of this process a rigorous formulation of these concepts is achieved and their formal treatment is made possible. Kreisel emphasizes that, contrary to the contemporaneous formalist trends, the role of formalization is auxiliary

[63] There is no indication how correctness is to be assessed.

[64] Gödel held similar views about the role of analyzing intuitive concepts. See p. 140 of his (1944) and Chapter 7 of Hao Wang's (1987).

here, what is important is the possibility and usefulness of the informally rigorous analysis of intuitive notions:[65]

> What the 'old fashioned' idea assumes is quite simply that the intuitive notions are *significant*, be it in the external world or in thought (and a *precise* formulation of what is significant in a subject is the result, not a starting point of research into that subject). (1967b, p. 138)

Once the significant notions have been found and formulated, the use of mathematical logic as a tool in foundational studies can begin:

> What logic does is to study notions [and rules of inference] which were previously not recognized at all, or, if recognized, only used heuristically, and not made an object of detailed study; among them traditional philosophic notions.[66] (1967b, p. 146)

The "philosophic notions" occur as a consequence of the different treatment and analysis of mathematical experience by the different philosophical traditions. As, according to Kreisel, "[t]he general ['old fashioned'] idea applies equally to the so-called realist conception of mathematics which supposes that these intuitive notions are related to the external world [...] and to the idealist conception which denies this or, at least, considers this relation inessential to mathematics." (p. 138) That is, the realist and idealist or intuitionistic traditions treat the "objects" in the foundational "data" in quite different ways. Kreisel himself supports realism towards mathematical objects in his (1967), where he also mentions that the difference between physical and mathematical evidence is usually seen as an argument against realism in philosophy of mathematics. Based on our current knowledge, Kreisel questions whether this distinction is so significant and should have such strong philosophical consequences.

As to different treatments of mathematical activity, some proposed a mechanistic or formalist theory of it, some a non-mechanistic one. Thus, according to Kreisel, the different philosophical views lead to different formalizations of mathematical experience. The difference between classical and intuitionistic mathematics, for instance, can be seen in that light.

We quoted already Kreisel's remark that the "precise formulation of what is significant in a subject is the result, not a starting point of research into that subject". That does not mean however that the formalizations together with technical developments do not have philosophical uses or consequences. Quite the contrary. Once formalizations are in place, some of them can be excluded or refuted based on formal or technical advances. Kreisel's example for this phenomenon from natural philosophy is Galileo's first theory of freely falling

[65] According to Kreisel the 'pragmatist' and 'formalist' traditions take seriously the experience of the physical world "but not what is counted as experience in traditional mental philosophy, namely insights into such intuitive concepts as logical validity, mechanical process, elementary proof" (1967, p. 205).

[66] However, no concrete examples are mentioned.

bodies which was rejected as the formulation led to the consequence that bodies in rest would never start to fall. For the philosophy of mathematics he observed:

> Also I believe (though this view is not shared widely) that the decision between different traditional positions, for instance realism and idealism, will be greatly helped by technical development: their formulations are so rudimentary that our present experience does not allow us to decide between them nor do they tell us what new experience to look for. (1967, p. 206)

This stance of Kreisel is not surprising, as for him "[t]he general problem of foundations is to analyze this experience taken as a *whole* or, at least, large portions of it." (1965, p. 95) That is, the aim of foundational studies is not only to find and formalize the significant properties of isolated notions, but to provide a theory of mathematical experience and mathematical practice. As a consequence foundational work should explain even quite specific phenomena of our mathematical experience:

> Suppose one wants to explain why a certain question [in mathematics] happens to be open; one guesses a formal system, i.e. properties of the intuitive notion, which mathematicians are likely to use; one supports this guess by showing that current mathematics follows from these axioms, and one explains the situation by showing that this question is not decided in the formal system (I personally like this sort of thing). (1967b, pp. 146-147)

It is clear that this requires very detailed analyses of "current mathematics" and "mathematical experience". As it will be important for the discussion of Church's Thesis, let us turn to Kreisel's description of what is required from such an analysis in the case of mathematical reasoning, namely, a "genetic theory".[67]

Genetic Theories

Genetic theories of mathematical reasoning will play an essential role in the discussion of recursive progressions and Church's Thesis. When discussing those issues, Kreisel frequently uses the "genetic theory of proofs" as an example and for comparisons. Hence the following overview focuses mostly on genetic theories of proofs and mentions genetic theories of reasoning only occasionally as those will be discussed in detail below.

"[T]he familiar aim of finding an all-encompassing formal system F for the whole of mathematics", Kreisel claims, was intended at the beginning of the 20th century as a genetic theory of proofs. What he means by it is more precisely expressed in the following:

[67] Kreisel uses the phrase "genetic theory" only in his (1972). However, in his earlier papers it is quite simple to identify the same issues based on appearance of other phrases, see footnote 68 for an example.

> The hypothetical formal system F was intended as a genetic theory of proofs. The old aim was not merely to find an F such that every theorem (in the language of F) which can be proved at all should also possess *some* derivation in F, but we should be able to see *how to get from an intuitive proof to its formalization* (in F). The latter was to be determined by the former modulo 'trivial' congruences.[68] (1972, p. 316)

However, it is remarked, "that *there is no convincing genetic theory even of those proofs which establish theorems of first order logic*".[69]

At places where Kreisel talks about theories of mathematical reasoning connected to formal proofs he mentions as a defect of those theories that they do not say anything "about the *actual* choice of proofs" (1967, p. 267). It is not clear from his descriptions, but in some cases he seems to expect from such theories that they predict which proof mathematicians would pick for a certain theorem. As an example he mentions that even in propositional logic, a simple theorem like $(p \wedge \neg p) \to (p \to p)$ can have "obviously different proofs [...] one using $p \to p$ and $q \to (r \to q)$ with $q = p \to p$ and $r = p \wedge \neg p$, the other using $(p \wedge \neg p) \to s$ with $s = p \to p$." (1971, p. 178)

It is not clear why the theory should give an account for the "actual" choice of mathematicians as it is not even claimed that this choice is homogeneous. Furthermore, if any of the above informal proofs are presented it is clear how to formalize them "modulo 'trivial' congruences".

Kreisel's discussion of Gödel's Incompleteness Theorem in this context provides further details about his ideas or expectations for a genetic theory. It shows that he wishes for a rather tight correspondence between actual mathematical experience and the formal system representing it:

> The mere existence of *some* sentence, even of low complexity such as Π^0_1, which is formally undecided in F, would not establish the inadequacy of F as a genetic theory of (actual) mathematical reasoning; number theory is full of open problems $\in \Pi^0_1$! (1972, p. 322)

Those currently open problems might be open due to "the limits of mathematical reasoning" (p. 316). Thus, for current genetic theories it is not a problem if those formulas are undecidable within the theory. Furthermore, if it turned out that they are unsolvable by mathematical reasoning, then they should be undecided in a genetic theory of proofs as well. This strong reading of the above quote is supported by Kreisel's interpretation of recursive progressions as genetic theories. As recursive progressions will be discussed later, I only quote here a short, relevant part without describing its context in detail: Accepting

[68] This phrasing of moving from an informal treatment to a formal one "modulo 'trivial' congruences" is present in Kreisel's earlier writings, where he talks about theories of proofs or reasoning without mentioning the term "genetic".

[69] For example, Kreisel points to an earlier footnote of his, namely (1971b, p. 113, fn. 1). As the context is different there and is not linked to questions discussed in his (1972), it is not immediately clear what kind of relevance that footnote bears for current issues, if it is relevant at all.

particular progressions as genetic theories of human reasoning, would amount to accepting "the assumption that all number theoretic total definitions formalized in this last progression are h[umanly]-effective." (p. 324) That is, everything that can be formally treated in a genetic theory should correspond to human capabilities.

These expectations for genetic theories require quite detailed analyses. Unfortunately, Kreisel's comments are rather scattered. In the following I list his phrases to describe such analyses. These phrases appear in describing genetic theories of any kind, not just of proofs. However, as Kreisel talks in these places about analyzing mathematical experience, these can be seen as part of his general philosophical treatment of our experience which leads to genetic theories at the end.

Although such a tight correspondence between the genetic theory and our experience is required by Kreisel, he says that:

> Any such theory would seem to need an *idealization far removed from our ordinary experience* (of human performances in mathematics). (p. 317)

This seeming incoherence is resolved a bit if we look at the role of the analysis in the following way:

> The rules and proofs are of course involved in the mental (psychological) experience of mathematical activity. But the theories we try to set up are not about this experience as it presents itself to us, they concern its logically significant aspects. (1970, p. 123)

The meaning of "logically significant aspects" is further explained by this analogy:

> [L]et us [...] compare our problems with those arising in mechanics, say. The objects considered in this science are not the (material) bodies of physical experience as they present themselves to our senses. To apply mechanics we have to determine the mechanically significant data in an empirical situation (mass of the body rather than, say, its colour, electric charges if there is electric interaction etc.) Indeed, we need the theoretical notions to tell us which data are significant, but we do not have a recipe for stepping from crude physical experience to those data. (1970, pp. 123-124)

Kreisel admits here that at the moment we have no systematic methods how to analyze our experience, how to find the significant aspects, not even in the case of mechanics. In addition, the above quote suggests eliminating or minimizing the role of psychological experience in the analysis. However, elsewhere he puts more weight on psychological features:

> [O]ne of the main purposes of the analysis is to restrict the notion of thinking subject so as to eliminate *accidental* psychological elements, yet to exploit essential ones. (1967b, p. 159)

Indeed, here the 'exploitation' of the essential "psychological elements" is suggested.

It is not clear from these descriptions how much role the psychological elements of our experience should have. Elsewhere Kreisel says that the "general problem of foundations is to analyze this [mathematical] experience as a *whole* or, at least, large portions of it" (1965, p. 95), and he talks about "*facts of actual intellectual experience*" (1967b, p. 141), and mentions "immediate experience" (1967, p. 201). Even if 'large portions of the experience as a whole' are understood as the experience without the accidental psychological features and can be seen to be in accord with or at least not in contradiction with the previous statements, it is clear that Kreisel's description is ambiguous.

This ambiguity is also transparent from Kreisel's always changing description of what kind of "data" should be analyzed. The mathematical experience of thinking and reasoning is considered as mental, abstract, and frequently as a psychological phenomenon. The different kinds of relevant "data" Kreisel mentions in connection to reasoning are: things on the "phenomenalistic level" (1965, p. 96, fn. 1), "psychological facts" (1967, pp. 218-219), "psycho-physical" "facts", "phenomena," and "nature of reasoning" (1967, pp. 268-269), "physical facts" and "physical behaviour" on the level of the "higher nervous system" (1967, p. 226). These different descriptions do not contradict each other, but Kreisel does not provide any explanation what he actually means by them, whether they overlap or complement each other, whether at least one or all of these have to be provided for a genetic theory.

Although failing to provide a clear and coherent picture of what a genetic theory should look like, one might be able to decide whether to accept a particular theory when it is provided. That is, once a concrete genetic theory is described in detail we might be able to see it as fulfilling these currently ambiguous criteria. At the same time such an example could advance our understanding of the notion of genetic theory in general as well.

Kreisel and Church's Thesis

With the above picture of Kreisel's general philosophy of mathematics in mind, how can Church's Thesis be analyzed? For, already its name, i.e. that it is a 'thesis', suggests that many believe that it cannot be treated formally as it identifies an informal notion with a mathematically precise one. However, we saw that Kreisel believes that the analysis of intuitive notions with the help of informal rigor can lead to a rigorous, mathematical formulation of those notions. Indeed, he states that the delicate mathematical analysis "*refutes* beyond a shadow of doubt such familiar claims as that the thesis is not problematic or that it is, in principle, not suitable for mathematical investigation." (1972, p. 317)

In the case of constructive or intuitionistic mathematics the question of Church's Thesis admits a formal and axiomatic treatment. In Kreisel's (1970, p. 122) Church's Thesis refers to the following formula, where f ranges over *constructive functions*, which are taken to be primitives:

$$\forall f \exists e \forall n \exists p [T(e, n, p) \wedge fn = U(p)]$$

where e, n, p range over natural numbers, and T and U denote Kleene's famous T predicate and his "result extracting" function.[70] And "for systems not containing variables for constructive (number theoretic) functions the axiom [...] is replaced [...] by the schema

$$\forall x \exists y \Re(x, y) \rightarrow \exists e \forall x \exists z [T(e, x, z) \wedge \Re(x, U(z))]$$

for all definable relations $\Re$ not containing free variables (parameters) for incompletely defined objects such as free choice sequences." (p. 126)

Once we have such formulations at hand the following questions can be asked intelligibly:

> *Are there simple syntactic conditions (on axioms) which ensure that the axioms, added to intuitionistic predicate calculus, are* (i) *consistent with Church's thesis or* (ii) *closed under Church's rule*[71] *?* (p. 127)

Here we see Kreisel's understanding of philosophy and mathematical logic as its tool working in practice: The notion of effectively-, humanly- or constructively calculable function is an informal one. The aim of philosophy or foundations of mathematics is to make the intuitions about this notion explicit and expressible by formulas within the language of a formal system. Then questions like the ones above can be asked and investigated by mathematical logic. Thus, philosophical or foundational questions might be solved with the aid of technical mathematical developments.

Of course Church's Thesis cannot be treated in the same way in case of classical number theoretic functions. One could similarly formally express a statement that every function is recursive with a second order formula. But to consider it as an axiom would be pointless, as we know that it would make the formal system inconsistent. Or, possibly, one could introduce the notion of *effective function* as primitive just as it was done with constructive functions.

Instead, Kreisel focuses on genetic theories of mathematical reasoning. There the question is whether these theories, representing our mathematical experience, are compatible with his interpretation of Church's Thesis. The technical apparatus of mathematical logic will help to reveal certain relevant properties of these genetic theories, especially in the case of recursive progressions.

In the next section I turn to Kreisel's criticism of Church's thesis in the classical setting. First I will discuss his criticisms of the Thesis and what he calls the *mechanistic* view before introducing transfinite progressions in the following subsection. Then with his ideas serving as a background, after introducing the technical details, I will survey his interpretation of the relevance of transfinite progressions for Church's Thesis.

[70] In connection to constructive functions Kreisel refers to (Bishop, 1967).

[71] Kreisel does not give any indication in the paper about there being any kind of difference between the 'thesis' and the 'rule.'

Church's Thesis and Recursive Functions

In this section I reconstruct Kreisel's criticisms of Church's Thesis in the classical case. The overview is based on his (1972).

For Kreisel "'effective' means [...] humanly performable and not only mechanical (instructions); in short, *h-effective* and not only *m-effective*." (1972, p. 314) While "Church's thesis is here taken to mean that every *total* function which has an h[umanly]-effective definition is recursive." (p. 315) Kreisel remarks, correctly, that this interpretation of Church's Thesis "is perhaps historically not quite accurate". (p. 314) Be that as it may, with the notions above in mind, according to Kreisel:

> Church's thesis for h-effective definitions belongs to the venerable subject of determining the nature and, especially, the limits of mathematical reasoning.[72] (p. 316)

According to Kreisel, Church's Thesis would have been confirmed if the attempts to formalize the whole of mathematics in one formal system at the beginning of the 20th century had been successful. As he puts it:

> Thus if the familiar aim of finding an all-encompassing formal system F for the whole of mathematics (or even the part dealing with number theoretic predicates) were realized, Church's thesis for h-effective definitions would be *established*. For any (F and any) *total* predicate defined by the formula $\mathscr{A}$ with a single free variable x, if $\mathscr{A}$ is decidable, that is, for each numeral n either $\vdash_F \mathscr{A}[x/n]$ or $\vdash_F \neg\mathscr{A}[x/n]$ then $\{n : \vdash_F \mathscr{A}[x/n]\}$ is recursive. (No further assumption is needed how we know that $\mathscr{A}$ is decidable.) (1972, p. 316)

However, since there can be no "all-encompassing formal system" and "*there is no convincing genetic theory even of those proofs which establish theorems of first order logic*", in order to establish Church's Thesis a genetic theory of humanly-effective definitions or a genetic theory of mathematical reasoning would be needed.

Interestingly, Kreisel accepts Turing's analysis of computability leading to a genetic theory, but to a genetic theory of *mechanistic*[73] definitions. Indeed, in this paper Kreisel interprets Turing's work in his (1936) as trying to establish a mechanistic theory of reasoning, that is, to show that human-effectiveness is the same as machine-effectiveness. At the point where Kreisel begins to discuss the notions of human- and machine-effectiveness, he remarks:

[72] This presentation of the Thesis and its relevance is very close to Post's understanding. Indeed, when talking about his characterization of solvability Post claimed that "the given set of instruments is in effect the only humanly possible set" (1965, p. 340) and interpreted the undecidability results as a "discovery in the limitations of the mathematicizing power of Homo Sapiens" (1936, p. 105).

[73] Kreisel uses the phrase 'mechanistic' only in his (1967) and not in his (1972), but as it describes the same phenomenon I will use it without further remark.

> The distinction was clearly recognized by the pioneers. It is presupposed in Turing's attempt (§9, p. 250, line 9) to *establish* that a 'machine can reproduce all steps that a human computer can perform'; presupposed, because one could not even raise the question of equivalence without understanding the distinction. (1972, p. 318)

I think this quote makes clear in the light of previous discussion of Turing's work that Kreisel misinterprets him. For, the 'human computer' represents a computing human that computes without any ingenuity, i.e. "mechanically". Thus it is not intended to capture the entire human-effectiveness or mathematical practice as Kreisel indicates it. As Turing's work and its context was discussed in detail previously, in the following I will regard Kreisel's interpretation misplaced without further arguments. He interpreted Turing's work differently in his earlier papers, e.g. his (1965).

Let us now return to Turing's analysis as a genetic theory of mechanical effectiveness. As Kreisel puts it:

> A paradigm of a genetic theory in the sense described (for m-effective definitions in place of proofs) is provided by Turing's analysis: each m-effective definition is *intentionally* [sic!] equal to some program for an 'idealized' computer, which I called 'Church's superthesis' in (1971, p. 177). (p. 316)

The reference the quote points to illuminates what Kreisel means when saying that Turing's analysis provides a genetic theory of mechanically-effective definitions:

> *superthesis*: to each mechanical rule or algorithm is assigned a more or less specific programme, modulo trivial conversions, which can be seen to define the same computation *process* as the rule. (1971, p. 177)

According to Kreisel "a moment's reflection shows" (1972, p. 316) that the conviction in the superthesis makes Church's Thesis accepted for mechanically-effective functions.

Now the question arises, how can Turing's analysis still suffice as a genetic theory of mechanically-effective definitions when it fails to analyze human-effectiveness, what it was intended to analyze according to Kreisel? This brings us to Kreisel's criticism of Turing's analysis. Namely:

> An [...] important error in Turing's argument consists in a kind of *petitio principii* assuming that the basic relations between (finite) codes of mental states must be mechanical;... (p. 319)

The mechanical relation between codes of mental states that the above quote refers to is Turing's requirement that the 'state of mind' of the 'human computer' and the symbol currently read by it determines uniquely its next 'state of mind' and action (i.e. writing/erasing a symbol on the current square or moving

to left/right). Thus, Kreisel indicates that Turing's analysis is circular, similarly to what is generally thought of Church's analysis in §7 of his (1936). Kreisel is right within his misinterpretation of Turing. That is, if someone argues for a mechanistic theory, then it is an inadmissible step to assume that the transition between 'states of minds' is mechanical (recursive or rudimentary) as it amounts to assuming the conclusion. Hence the application of this assumption immediately restricts the analysis to mechanical processes. And Kreisel agrees with the vast majority of logicians that as an analysis of mechanically effective processes, Turing's analysis does succeed.

Let us now take a closer look at Kreisel's criticism of the mechanistic view in general and then return to the interpretation of Turing's work. Understanding why Kreisel thinks these "attempts" fail to establish a 'genetic theory of humanly-effective definitions' will provide a deeper comprehension of his requirements for such a theory. In Kreisel's presentation of the mechanistic theory of reasoning, the mechanisms were claimed to appear already on the level of the nervous system:

> Probably the major attraction of formalization was that it suggested the possibility of a *mechanistic theory* of human reasoning, in particular, that the propositions [...] not only can be decided by means of the formal rules, but that something like repeated application of such rules is all that goes on even if we consciously think of reasoning differently; more precisely, that the higher nervous system consists of a *mechanism* whose behaviour is given by the formal rules, as an electronic computer is a mechanism whose physical behaviour realizes certain mechanical laws (the 'instructions' which it is given). (1967, p. 226)

But at the same time they fail even to give an analysis of how we choose and proceed in case of mathematical proofs:

> It is remarkable how little work was done on this even in areas, such as predicate logic, where the set of valid statements is recursively enumerable. The least one would have to do is to show that there is something mechanical about the *actual* choice of proofs, not only about the set of results: after all, one may ride to work on a camel or a donkey and get there: but this does not mean that a camel is a donkey. (1967, p. 267)

Again, according to Kreisel, it should not be just assumed that our reasoning is mechanical, as our psychological experience suggests the contrary. Moreover he is right that even if we are misled by our experience, assuming from the outset our reasoning to be mechanical is mistaken. Instead, it should be treated as one of the possibilities:

> Mathematical reasoning, except in the case of numerical computations, does not present itself to us as the execution of mechanical rules. Even where such a reasoning can be *re*presented, more or less

> adequately, by mechanical means, this possibility had to be *discovered*. (1970b, p. 21)

It seems that mathematical experience, how it "present[s] itself to us", and how it appears on the cognitive level are all important for Kreisel in providing a genetic theory of reasoning.

Kreisel's presentation of the mechanistic view might seem to be a bit overdrawn. I already remarked that he misinterprets Turing's work. Beyond that even the general picture of the mechanistic view seems to be overly simplistic and radical, lacking arguments to back it up. Although the topic is approached with more care recently and many people did so already in the 1960s, readers should remind themselves that at the time Kreisel wrote these articles the first wave of artificial intelligence research just culminated. That is, such exaggerated announcements of the initial successes were not unprecedented.[74]

Let us return now to Kreisel's criticism of Turing's analysis in more detail. It was already mentioned above that Kreisel saw a *petitio principii* in Turing's argument. He elaborates on what should be done in order to obtain a genetic theory of humanly-effective definitions:

> Quite obviously, coding (mental) states of the human computer is a much more delicate matter. More formally, even if we assume a coding by *finite* configurations, say by finite sets of words or by derivations (in a formal system or in a progression), the issue is this:
>
> What is the *arithmetic* character of the relations (between codes) induced by meaningful relations between the mental states considered? Are those also rudimentary? (1972, p. 319)

It is important to emphasize that this coding connects "mental acts" to natural numbers, i.e. to codes. This strengthens the view that the analysis required for a 'genetic theory of humanly-effective definitions' should involve psychological or cognitive analysis as well.

Kreisel provides an example where it turned out "how well we (sometimes) understand" the above coding relation. However, as we will see it confuses the picture of what is actually required from an analysis for a genetic theory of humanly-effective definitions. He points to the example with the following description:

[74] Interestingly, Kreisel refers only to Gandy's (1969) as one such example. However, although Gandy is imprecise in cases, he does not propagate any of the views of which Kreisel accuses him and he definitely does not propagate the mechanistic theory. For example on the comparison of brains with computers Gandy says that "it is now becoming clear that the brain organizes its computations quite differently from electronic machines." (p. 3) Moreover, he does not suggest that human reasoning or its experience seems to be mechanistic:

> Of course, the flexibility of the mind when faced with a problem is dazzling; it can select an appropriate known rule, create a new one, or work unsystematically. (p. 4)

> To avoid (quite unnecessary) doubts, the reader should immediately look at the standard example, on p. 124–125 of (1970), of an h-effective definition which actually uses the coding itself, that is, it involves the passage between a formal derivation (in Heyting's arithmetic) and the corresponding mental act, namely the proof expressed by the derivation. (1972, p. 320)

The reference in the above quote points to the following function, which is a "sort of example [that] is most natural if one wants to look at problematic aspects of the coding relation at all":

> **Basic example: constructive (prima facie) non-mechanical rules for number theoretic functions.** We consider a formal system such as Heyting's arithmetic HA which we have recognized to be constructively valid. Thus to each formal derivation (with Gödel number) n corresponds a constructive proof, say p_n. The rule f is given by cases:
>
> Case 1. If n is not a derivation of any closed formula of the form $\exists x\mathfrak{A}$ then $fn = 0$. Otherwise
>
> Case 2.1. $fn = 0$ if p_n does not provide a *specific* numerical instance satisfying $\mathfrak{A}$, e.g. if $\exists x\mathfrak{A}$ has been inferred from $\forall x\mathfrak{A}$;
>
> Case 2.2. $fn = x+1$ if x is the number verifying $\mathfrak{A}$ which is provided by p_n. (1970, p. 124)

From the title, "constructive (prima facie) non-mechanical" rule, it follows that Kreisel thought that the function defined above is not trivially mechanizable, and he mentions that "clearly, no Turing machine will react to this rule as it stands." In his (1972) he says that "it was good luck that only two years after [this function] was presented, one was able to give an *equivalent* explicit, in fact, m-effective definition by use of normalization technique" (p. 320). As a reference, Kreisel points to §1b of his (1971b). However, no equivalent mechanical definition is provided there. As this example is illustrative of Kreisel's discussion of the issue; so I describe the context in which it appears in his (1971b).

The outset is rather similar:

> The general nature of our problem is quite clear. Consider *formal rules* which are indeed to formalize certain *proofs*; in other words, we have syntactic objects, derivations, d which represent or describe mental acts $\bar{d}$, the proofs (which carry conviction). (p. 111)

Thus a connection or "mapping" is sought between mental acts and their formal representations. More precisely:

> Given a property P of or a relation R between proofs our task is to find relations P_F, resp. R_F such that for all d [derivation] of our formal system
>
> $$P_F(d) \text{ iff } P(\bar{d}) \text{ and } R_F(d, d') \text{ iff } R(\bar{d}, \bar{d'}).$$

> For exposition, we shall reverse this procedure, and first describe some formal relations P_F, R_F which will then be used to state the facts about the objects of principal interest, namely properties and relations of proofs. (p. 111)

Here, again, we see Kreisel's general philosophical approach working. His aim here is to use normal derivations as canonical representations of proofs, i.e. of mental acts. The new results concerning normalization processes that he alludes to ensure that even if normalization steps are applied in a different order they lead to the same normalized proof. He considers it as a fulfillment of a minimum formal requirement to use them as canonical representations. A further, informal requirement is that conversion steps have to preserve the identity of proofs, i.e. of mental acts. According to him, inspection shows that the conversion steps used in the new results (contracting the introduction of a logical symbol immediately followed by its elimination) "evidently satisfy" this requirement.[75] (p. 112)

Then, in §1b these results are applied to the function above. Here Kreisel talks about functions of the form that match with his 'Basic example':

> the definition $\bar{t}$ is provided by the proof $\bar{d}$ of the existential assertion $\overline{\exists x A}$ (p. 114)

Here $\bar{d}$ and $\bar{t}$ still denote the mental acts. In Kreisel's explanation:

> An important property of *normal derivations* is that they allow us to *read off* t *from a derivation* d *of* $\exists x A$; more precisely there is a mechanical method of obtaining the term t from d. Obviously there is a mechanical method of deciding whether d is a derivation of an existential formula. (p. 114)

Hence, Kreisel indicated that this function is one of the examples of "how well we (sometimes) understand the [coding] relation" because an equivalent, mechanically-effective definition of it was given.

However, it is not clear from this description what we learned about the connection between proofs, i.e. mental acts, and their codes or representations. Kreisel himself admits that the procedure of studying proofs is reversed and first he describes formal relations "which will then be used to state the facts about [...] proofs." Although he claims that it is done only "for exposition", no detailed analysis of the mental acts is ever delivered. Moreover, if in his 'Basic example' the correspondence between the constructive proofs p_n and formal derivations (with Gödel number) n was problematic, it still should be considered problematic. For, what we learned from the normalization proofs is that certain formal derivations with different Gödel numbers lead to the same normalized derivation. But these results, although important, do not illuminate the correspondence we were originally interested in.

[75] At the same time Kreisel remarks that "inspection shows that many cut elimination procedures for calculi of sequents do not obviously satisfy the informal requirement" (p. 113).

Even if the above function and its mechanization is accepted as an example of "how well we (sometimes) understand" the coding relation between mental acts, did the above function capture anything from our mathematical practice or experience? Does this count as a psychological or cognitive analysis of any sort? We saw that sometimes Kreisel seems to expect only idealizations "far removed from our ordinary experience". Even so, did we learn anything about humanly-effective definitions in this context?

It is also astonishing that Kreisel agrees with Gödel's critique of Turing's argument, claiming that the "assumption about distinguishable states of mind are problematic"[76] (p. 318) but Kreisel himself does not provide any description of 'states of minds' or 'mental acts', he only speaks about codes of whole proofs. (It should be noted that Turing himself actually eliminates the notion of 'state of mind' from his analysis and replaces it with its "physical counterpart", with a "note of instructions". (Turing, 1936, p. 253))

The above function seems to be problematic from another point of view. Kreisel accused the mechanist view of not even trying to "show that there is something mechanical about the *actual* choice of proofs, not only about the set of results". (1967, p. 267) But the function and its coding above does not say anything about the "actual choice of proofs" either.

Thus Kreisel criticized Turing's analysis, and it is not immediate that what he proposed instead was relevant for the issues he raised. The next subsections describe the results on recursive progressions and their relevance for Church's Thesis in Kreisel's interpretation.

Recursive Progressions

Turing, in his PhD (1938) written under the direction of Church, was interested in the possibility of extending formal systems. He considered an infinite sequence of extending logics or what he called *ordinal logics*.[77] This idea was later picked up as a research topic in the 1960s, primarily by Feferman. Below I summarize the main ideas and results, following Rathjen and Sieg's (2015).

Let T be a theory which is consistent but is shown to be incomplete by Gödel's theorem. Then there are statements that are not provable in T, but can be joined to it to make it "less incomplete," resulting in a new theory T'. Usually the following formulas, so called *reflection principles*, are taken into consideration:

(R_1) $\qquad T' = T + Con(T)$

(R_2) $\qquad T' = T + \{Pr_T(\ulcorner \phi \urcorner) \longrightarrow \phi \mid \phi \; closed\}$

(R_3) $\qquad T' = T + \{\forall x Pr_T(\ulcorner \phi(\bar{x}) \urcorner) \longrightarrow \forall x \phi(x) \mid all \; \phi(x) \; with \; at \; most \; x \; free\}$

[76] Such views of Gödel can be found in his (1972) and (1972a).

[77] Feferman's (1988) gives an accessible overview of Turing's work on ordinal logics and its historical context. In the *Technical Appendix* of the paper Feferman recasts Turing's work, originally done in the terms of λ-calculus, in recursion theoretical terminology.

where $Con(T)$ is the formula expressing the consistency of T, Pr_T is the provability predicate of T, $\ulcorner\phi\urcorner$ is the Gödel number of ϕ, and $\bar{x}$ is a numeral.

Then, consider the following sequence of theories $T = T_0 \subseteq T_1 \subseteq T_2 \subseteq \ldots$ where $T_{i+1} = T_i'$ for successor ordinals and $T_\lambda = \cup_{\alpha<\lambda} T_\alpha$ for limit ordinals. "However, since one needs to be able to express the provability predicate for T_λ in the language of T_λ itself one cannot simply use set-theoretic ordinals. Moreover, as one really wants theories whose axioms are effectively presented, i.e., can be enumerated by a recursive function [...] one has to deal with ordinals in an effective way." (p. 30) For this purpose Kleene's system of ordinal notations, "Kleene's O", can be used, where numerical codes are used to denote the constructive ordinals.

The ordinal notation was first introduced by Church and Kleene in their (1937) which used expressions of the λ-calculus instead of numericals. Kleene's O is an equivalent definition in terms of numericals, introduced in his (1938) and (1944). Here, again, I follow Rathjen and Sieg.

Kleene used $suc(a) = 2^a$ as a notation for successor ordinals and $lim(a) = 3 \cdot 5^a$ for limit ordinals. The ordinal notations O and the partial ordering $<_O$ between such notations and the ordinal $|a|$ denoted by $a \in O$ are defined simultaneously as follows:

(i) $0 \in O$ and $|0| = 0$,

(ii) if $a \in O$ then $suc(a) = 2^a \in O$, $a <_O suc(a)$ and $|suc(a)| = |a| + 1$,

(iii) if e is an index of a total recursive function and
$\{e\}(n) <_O \{e\}(n+1)$ for all $n \in \mathbb{N}$,
then $lim(e) = 3 \cdot 5^a \in O$, and $|lim(e)| = sup\{|\{e\}(n)| \mid n \in \mathbb{N}\}$,

(iv) if $a <_O b$ and $b <_O c$ then $a <_O c$.

The first non-constructive ordinal, i.e. the first ordinal that does not have a code in Kleene's O is denoted by ω_1.[78]

Completeness results were proven for such sequences of theories, so called progressions, already by Turing in his PhD, and were strengthened further by Feferman in his (1962). From the above definition of constructive ordinals it is clear that the ordinal notation is not unique. Thus to achieve completeness results one needs to choose a *unique notation in* O, a so called *path through* O. A set Z is called a path through O if:

(i) $Z \subseteq O$,

(ii) $c, d \in Z \longrightarrow c \leqq_O d \vee d \leqq_O c$,

(iii) $c \in Z \;\&\; d \leqq_O c \longrightarrow d \in Z$,

(iv) $|Z| = \omega_1$.[79]

[78] Sometimes it is also denoted by ω^{CK} or ω_1^{CK} in honor of Church and Kleene.
[79] If $|Z| \leqq \omega_1$ then Z is called a path *within* O.

Let Tr be the set of all true sentences of first order arithmetic, and Tr_1 be the set of all true sentences in the form $\forall x\phi(x)$, then of course $Tr_1 \subseteq \Pi^0_1$. The completeness result achieved by Feferman says informally the following:

For a suitable recursive progression of theories T_d, there is a path P through O which is recursive in O such that

$$Tr = \cup_{d\in O} T_d = \cup_{d\in P} T_d.^{80}$$

"However", as Feferman and Spector remark in their (1962, p. 384), "because of a non-uniqueness result which holds for many recursive progressions – that for each $c \in O$, $|c| \geqq \omega + 1$, there exists a $d \in O$ with $|d| = |c|$ and" $T_d \neq T_c$ – "it was expected that the above completeness theorem would not necessarily transfer to arbitrary paths through O." Indeed, "*there is a wide class of paths Z for which*" $\cup_{d\in Z} T_d$ "*is incomplete even with respect to the sentences of* Tr_1." More precisely, on page 385, they prove the following Incompleteness theorem:

If Z is a path through O and $Z \in \Pi^1_1$ then $Tr_1 \nsubseteq \cup_{d\in Z} T_d$.

This result was further strengthened by Kreisel in his (1972), based on the ideas of Feferman and Spector:

> If each numerical instance $A[x/n]$ of the formula A (with the free numerical variable x) is decided in a recursive progression along a Π^1_1-path then the set $\{n : \vdash A[x/n]\}$ is recursive. (Kreisel, 1972, p. 311)

Church's Thesis and Recursive Progressions

Kreisel considered Turing's work on ordinal logics as a "*model of mathematical reasoning*" (1972, p. 324), as a genetic theory of reasoning. Recursive progressions are "variants" of ordinal logics, where "Feferman [...] chose *paths* through O for the indexing expressing a kind of single-minded progress of mathematics" (p. 324) As a consequence Kreisel saw these progressions as relevant for "our current knowledge about Church's thesis." (p. 311) First I will take a look at Kreisel's views on whether the results concerning progressions can be reconciled with the thesis. Then the question of their adequacy as 'genetic theories' of mathematical reasoning will be discussed.

Kreisel interprets the completeness and incompleteness results of progressions in section (b)(ii)(α) *Recursive progressions and Church's thesis* (pp. 324–325). First he interprets Feferman's (and Turing's) completeness result. As it was stated above, for certain paths P recursive in O, the progression along P is complete for arithmetic. According to Kreisel:

> As a corollary, Church's thesis (just for total functions) is *incompatible* with the assumption that all number theoretic total definitions

[80] The informal summary of the result is based on (Feferman and Spector, 1962, p. 384)

> formalized in this [...] progression are h-effective. (italics by me, p. 324)

Thus, according to Kreisel, if a particular progression of the kind above serves as a genetic theory of mathematical reasoning, then every total definition that can be formalized in the progression is considered to be humanly-effective. Kreisel does not give any explanation of why that assumption is being made. Once the assumption is made, then, due to the completeness of the progression we know that there are total definitions formalizable in the progression, but not recursive. Hence, such progressions are not compatible with Church's Thesis, as it states that every humanly-effective definition is recursive.

On the other hand, in case of the incompleteness results:

> For Π^1_1-paths π and every consistent recursive extension principle [reflection principles] Church's thesis (for *total* functions!) holds [...] provided every h-effective definition can be formulated in the language of the progression and every computation from these definitions can be formalized. (p. 325)

It is immediately added that "[i]t is not claimed that the assumptions above are plausible." Considering the progressions and paths therein it also has to be mentioned that "[i]n the case of paths P, there is no convincing evidence that we can *recognize* the truth of $n \in P$. In the case of paths π, if we can really always recognize the truth of $n \in \pi$, there is no clear reason why we cannot do so for other kinds of paths." (p. 325)

What can then be said about the status of Church's Thesis in the light of these results? According to Kreisel the state of affairs can be summarized in the following way:

> *Unless it can be shown that the progression on P is not included in any* (legitimate) *model of mathematical reasoning, we cannot establish Church's thesis* (for h-effective definitions). And *unless it can be shown that each recursive progression on a Π^1_1-path is inadequate* (as a model for mathematical reasoning) *we cannot refute Church's thesis.* (p. 325)

Although Kreisel claims that this is a "better summary of the state of affairs", it is still rather hard to follow. In the following the above quote is unfolded to get a better grasp of his view.

The first part says that if it can be shown that the progression on P is excluded from the set of reasonable candidates for genetic theories of reasoning, then there is a chance to establish Church's Thesis. For, to put it more similarly to the original phrasing, as long as we cannot exclude the progression on P from the circle of reasonable candidates for genetic theories, we cannot establish Church's Thesis. That is, this progression is incompatible with the thesis, hence as long as the progression on P is included among the candidates, one can be skeptical about the truth of the thesis, as P might be a counter-example.

Hence, to establish Church's Thesis it is necessary, but obviously not sufficient, to exclude these progressions from the models of mathematical reasoning.

The second part says that if progressions on a Π^1_1-path can be excluded from the set of reasonable candidates for genetic theories of reasoning, then there is a chance to refute Church's Thesis. For, recursive progressions on a Π^1_1-path are compatible with the Thesis, in order to refute the Thesis one has to show all of them to be inadequate as genetic theories. Of course it does not mean that if the inadequacy of these progressions is indeed showed, then the thesis is immediately refuted. Rather, the case is similar to the previous one. As long as there is at least one recursive progression on a Π^1_1-path, which is compatible with the thesis, that is included among the candidates of a genetic theory of mathematical reasoning, the thesis cannot be considered as refuted. Thus, again, it is a necessary but not sufficient criterion, but in this case to refute the thesis.

Based on the unusual phrasings, as Church's Thesis being incompatible with one and holding in another progression, the question arises, how does Kreisel exactly understand the Thesis in this context? The version presented by him on page 315, "Church's thesis is here taken to mean that every *total* function which has an h-effective definition is recursive", is a variant of the Thesis in the sense that it identifies an informal notion, i.e. functions which have h-effective definitions, with a mathematically precise one. However, throughout the *Introduction* and part I of the paper, Kreisel talks about *reckonable functions* instead of ones that have h-effective definitions. He says that his strengthening of Feferman and Spector's results (see above) "is needed to extend Church's thesis for *total* functions reckonable in formal systems to (total) functions reckonable in recursive progressions on Π^1_1 paths." (p. 311) It is an important difference, as reckonable functions are a formally defined class of functions for formal systems by Hilbert and Bernays (see Section 3.5 of Sieg's (2009) for further details about them). Kreisel claims that "a simple verification [will be given] that reckonability in a progression π is a special case of enumeration reducibility π" (p. 311) Church's Thesis understood this way becomes a mathematical statement that is provable or disprovable about a particular theory. With this interpretation of the Thesis Kreisel's use of words as "holds in a theory" or "incompatible with a theory" are more understandable. At the same time, it is clearly a different kind of statement than the usual understanding of Church's Thesis.

Let us not forget that Kreisel agrees with a general understanding of the relevance of the Theses, namely that "Church's thesis for h-effective definitions belongs to the venerable subject of determining the nature and, especially, the limits of mathematical reasoning." (p. 316) If the Thesis is understood as identifying reckonable functions with recursive ones, then in order to make the results indeed relevant for the "nature" and "limits of mathematical reasoning" progressions have to be shown to be strong theories of mathematical reasoning with a notion of reckonability that corresponds to "h-effective definitions". It would be in accord with Kreisel's general philosophy, where the informal analysis leads to correct formulation of notions, which, in turn, make the application of technical results possible and lead to philosophical conclusions.

Although Kreisel mentions in the *Introduction* that "a simple verification [will be given] that reckonability in a progression π is a special case of enumeration reducibility π" (p. 311), he only provides the following explanation:

> (α) if the set X is *reckoned* in our progression by the formula $\mathscr{P}$ [with a single free (numerical) variable] above then X is recursive.
>
> It is clear that enumeration and not Turing reducibility is relevant. When using the progression to decide between $\mathscr{P}[x/n]$ and $\neg\mathscr{P}[x/n]$ we use only elements of π, not of its complement (which could also be used in Turing reducibility). Indeed, whenever we consider inductively defined sets X such as r.e. sets of theorems in formal systems or Π^1_1 sets of theorems in a progression on Π^1_1-paths, we think of X as distinguished from its complement by asymmetry: we know the members of X 'eventually' and we may 'never' know whether an object is not a member. Turing reducibility ignores this asymmetry. (p. 314)

It indeed makes enumeration reducibility more appealing than Turing reducibility. However, it does not provide an analysis or a positive argument of why enumeration reducibility should be identified with reckonability.

In terms of genetic theories, recursion theory was already rejected as the genetic theory of mathematical reasoning by Kreisel. Are these progressions better alternatives in any way, and if they are, why? Kreisel admits that "the progressions considered [...] cannot be claimed to provide realistic models of mathematical reasoning". However, at the same time, "they are useful for *illustrations* and other pedagogic purposes." (p. 325) Next I will look into these properties to get a further grasp of why Kreisel considers progressions to be slightly more appropriate candidates for genetic theories of mathematical reasoning, and, also what problems they have.

First of all, in case of any particular formal system (one formal system, not a progression of theories), "we could not have *mathematical* evidence[81] for [its] adequacy". For, according to Kreisel (p. 322) a particular formal system F is not shown to be inadequate for a "genetic theory of (actual) mathematical reasoning" just because it is incomplete, as "number theory is full of open problems $\in \Pi^0_1$." These formal systems are inadequate because they fail to satisfy the "*minimum* requirement", that is, to be "demonstrably sound, at least for Π^0_1 sentences, or as one sometimes says, the *reflection principle* with respect to Π^0_1 sentences must be provable for F." Thus, formal systems are inadequate as it is a consequence of Gödel's First Incompleteness Theorem that the reflection principle concerning the Gödel sentence cannot be proved in F. At the same time this also means that recursive progressions do not possess this inadequacy as they, by construction, do contain reflection principles as axioms.

Let us continue with an example of Kreisel's where progressions are used to model mathematical practice. As his example he considers Brouwer's 'thinking subject', which is not connected to the issues that are considered here. The

[81] Although there could be some non-mathematical or empirical evidence for them.

overview below is just for illustration of his treatment of the matter and it is left fragmentary.

An ω-ordering is used to deal with "the body of mathematical evidence as the 'thinking subject' or, equivalently, the idealized mathematician proceeds in time." (p. 325) Then:

> [I]f he starts with first order number theory at stage 0, he will immediately *interpret* some of his formal theorems as proofs about (possible) proofs at stages $< \epsilon_0$. For the present model the ω-ordering of 'stages' would be given by a sequence of elements $e_1, e_2, ...$ on the path cofinal with the whole path. (p. 325)

Here, again, Kreisel does not give any further psychological motivation, explanation, or any kind of theory of the "(actual) mathematical reasoning."

However, he admits that this model "leaves open quite basic questions", namely:

> Is only conscious knowledge involved or [sic!] all he 'can' know? Are they stages in the realization of some given masterplan, that is of given possibilities of the mathematical imagination? Or is some (internal or external) choice among these possibilities significant? (p. 326)

This list of open or untouched questions is surprising in the light of Kreisel's previous treatment of mathematical reasoning. Although in the beginning it seemed that he would require a serious psychological analysis of mathematical reasoning, the actual discussion of those matters was rather shallow. Indeed, he restricted the scope of such analyses by claiming "the theories we try to set up are not about this [mathematical] experience as it presents itself to us, they concern its logically significant aspects." (1970, p. 123) As a result, it was hard to find anything concerning mathematical experience and its psychological analysis in his models.

Mentioning "conscious knowledge" and "mathematical imagination" seems to be incoherent or problematic in at least three ways. First, as mentioned above, it is not obvious to what extent these belong to the "logically significant aspects" of mathematical experience. That is not to say that these should not be accounted for. It is just not clear how these could fit into Kreisel's previous picture of analyzing mathematical experience. Second, his 'basic example' of a function given by a "constructive (prima facie) non-mechanical" rule does not seem to address these questions at all. As a reminder, that function was an example of "how well we (sometimes) understand the relation" between mental states (and their codes). However, there is no place to talk about conscious or unconscious knowledge, no place for imagination and choices to be made while realizing a masterplan. It is not clear that if the above questions are relevant for a genetic theory of mathematical reasoning, then how can that function claimed to be adequate, moreover showing our understanding of these delicate matters.

Finally, the question of the relevance of the progressions. Kreisel was forthright in saying that "the progressions considered [...] cannot be claimed to provide

realistic models of mathematical reasoning" but in light of the above deeper psychological questions it is not clear how they could be relevant at all for a genetic theory addressing these questions. It also makes his summary of the state of affairs and their relevance for Church's Thesis questionable. Either it is not clear what it would mean for a progression to be included in or being inadequate for a supposedly very complex genetic theory of mathematical reasoning that addresses all these issues. Or, more likely, it shows the plain irrelevance of the progressions for theories of mathematical reasoning and for Church's Thesis. Moreover it brings out how weak the necessary and insufficient criteria in Kreisel's summary of the state of the affairs are. It seems that he himself was aware of this as he briefly says:

> The classes of paths considered are characterized by purely syntactic conditions with little (known) virtue except that they allow us to prove something (of dubious relevance). (1972, p. 325)

To sum up, we can say the following. Kreisel's general attitude is against the acceptance of Church's Thesis. He addressed the question in a mathematical framework, interpreting recursion theory and results about recursive progressions as theories of mathematical reasoning. However, according to Kreisel, current results are definitely not sufficient to settle the question. It has to be mentioned that it is not clear whether these results are indeed relevant for the evaluation of Church's Thesis. Throughout his discussion a psychological analysis was mentioned and sometimes even required. At the same time it is not clear what kind of analysis he was looking for, as in cases it appears to be restricted and minimal, when he speaks about the "logically significant" parts of the experience, while in other cases it seems to be deeper, when he mentions "conscious knowledge" and "imagination". In addition, his examples demonstrating our knowledge on these issues seem to lack the psychological analysis he required.

Bibliography

Akl, Selim. 2000. "The Design of Efficient Parallel Algorithms." In Blazewicz, Ecker, Plateau and Trystram (eds) Handbook on Parallel and Distributed Processing. Berlin: Springer Verlag, 13–91.

Austerweil, Joseph and Thomas Griffiths. 2013. "A Nonparametric Bayesian Framework for Constructing Flexible Feature Representations." *Psychological Review* 120, no. 4: 817–851.

Bernstein, Ethan and Umesh Vazirani. 1997. "Quantum Complexity Theory." *SIAM Journal on Computing* 26, no. 5: 1411–1473.

Bishop, Errett. 1967. *Foundations of Constructive Analysis.* New York: Academic Press.

Black, Robert. 2000. "Proving Church's Thesis." *Philosophia Mathematica* 8, no. 3: 244–258.

Bramley, Neil, David Lagnado and Maarten Speekenbrink. 2014. "Conservative Forgetful Scholars: How People Learn Causal Structure Through Sequences of Interventions." *Journal of Experimental Psychology: Learning, Memory, and Cognition* 41, no. 3: 708–731.

Church, Alonzo. 1935. "An Unsolvable Problem of Elementary Number Theory. Preliminary Report." *Bulletin of the American Mathematical Society* 41: 332–333.

Church, Alonzo. 1936. "An Unsolvable Problem of Elementary Number Theory." *The American Journal of Mathematics* 58, no. 2: 345–363.

Church, Alonzo. 1937a. "Review of Turing's (1936)." *Journal of Symbolic Logic* 2, no. 1: 42–43.

Church, Alonzo and Stephen Kleene. 1937. "Formal Definitions in the Theory of Ordinal Numbers." *Fundamenta Mathematicae* 28: 11–21.

Claude, Christian, Solomon Marcus and Ionel Tevy. 1979. "The First Example of a Recursive Function which is not Primitive Recursive." *Historia Mathematica* 6, 380–384.

Coenen, Anna, Bob Rehder and Todd Gureckis. 2015. "Strategies to Intervene on Causal Systems are Adaptively Selected." *Cognitive Psychology* 79: 102–133.

Copeland, Jack and Oron Shagrir. 2007. "Physical Computation: How General are Gandy's Principles for Mechanisms?" *Minds and Machines* 17, no. 2: 217–231.

Cotogno, Paolo. 2003. "Hypercomputation and the Physical Church-Turing Thesis." *British Journal for the Philosophy of Science* 54: 181–223.

Culler, David et al. (1993). "LogP: Towards a Realistic Model of Parallel Computation." In *Proceedings of the Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming.* 1–12.

Damasio, Antonio. 1994. *Descartes' Error.* New York: A Grosset/Putnam Book.

Danks, David. 2014. *Unifying the Mind.* Cambridge, Massachusetts: The MIT Press.

Davis, Martin. 1982. "Why Gödel Didn't Have Church's Thesis." *Information and Control* 54, no. 1-2: 3–24.

Davis, Martin. 2004. "The Myth of Hypercomputation," In Teuscher, Christof (ed) *Alan Turing: Life and Legacy of a Great Thinker.* Berlin: Springer Verlag, 195–211.

Davis, Martin and Wilfried Sieg. 2015. "Conceptual Confluence in 1936: Post and Turing." In Sommaruga, Giovanni and Thomas Strahm (eds) *Turing's Revolution The Impact of His Ideas about Computability.* Springer International Publishing. 3–27.

Dedekind, Richard. 1888. "Was sind und was sollen die Zahlen?" In Dedekind *Essays on the Theory of Numbers.* New York: Dover Publications. 1963. 29–115.

De Pisapia, Nicola. 2000. *Gandy Machines: an Abstract Model of Parallel Computation for Turing Machines, the Game of Life, and Artificial Neural Networks.* Masters Thesis, Carnegie Mellon University.

Dershowitz, Naum and Yuri Gurevich. 2008. "A Natural Axiomatization of Computability and Proof of Church's Thesis." *The Bulletin of Symbolic Logic* 14, no. 3: 299–350.

Feferman, Solomon. 1962. "Transfinite Recursive Progressions of Axiomatic Theories." *The Journal of Symbolic Logic* 27, no. 3: 259–316.

Feferman, Solomon. 1988. "Turing in the Land of $O(z)$." In Herken, Rolf (ed) *The Universal Turing Machine, A Half-Century Survey.* Wien: Springer Verlag. 103–134.

Feferman, Solomon and Clifford Spector. 1962. "Incompleteness Along Paths in Progressions of Theories." *The Journal of Symbolic Logic* 27, no. 4: 383–390.

Fernbach, Philip and Steven Sloman. 2009. "Causal Learning with Local Computations." *Journal of Experimental Psychology: Learning, Memory, and Cognition* 35, no. 3: 678–693.

Folina, Janet. 1998. "Church's Thesis: Prelude to a Proof." *Philosophia Mathematica* 6, no. 3: 302–323.

Folina, Janet. 2006. "Church's Thesis and the Variety of Mathematical Justifications." In Olszewski, Adam, Jan Wolenski and Robert Janusz (eds) *Church's Thesis After 70 Years.* Frankfurt: Ontos Verlag. 220–241.

Fortune, Steven and James Wyllie. 1978. "Parallelism in Random Access Machines." In Proceedings of the 10th ACM Symposium on Theory of Computing. 114–118.

Frijda, Nico. 1986. *The Emotions.* Cambridge: Cambridge University Press.

Gandy, Robin. 1969. "The Concept of Computability." In Harré, Romano (ed) *Scientific Thought 1900-1960: A Selective Survey.* Oxford: Calderon Press. 1–4.

Gandy, Robin. 1980. "Church's Thesis and Principles for Mechanisms." In Barwise, Keisler, and Kunen (eds) *The Kleene Symposium.* Amsterdam: North–Holland Publishing Company. 123–148.

Gandy, Robin. 1988. "The Confluence of Ideas in 1936." In Herken, Rolf (ed) *The Universal Turing Machine – A Half-Century Survey.* Oxford: Oxford University Press, 55–111.

Gardner, Martin. 1970. "The Fantastic Combinations of John Conway's New Solitaire Game 'Life'." *American Scientific* 223: 120–123.

Gentner, Dedre and Linsey Smith. 2013. "Analogical Learning and Reasoning." In Reisberg, Daniel (ed) *The Oxford Handbook of Cognitive Psychology.* New York: Oxford University Press: 668–681.

Goldwater, Micah and Dedre Gentner. 2015. "On the acquisition of abstract knowledge: Structural alignment and explication in learning causal system categories." *Cognition* 137: 137–153.

Gödel, Kurt. 1931. "On Formally Undecidable Proposition of Principia Mathematica and Related Systems I." In Feferman, Solomon (ed) *Kurt Gödel. Collected Works. Vol. 1.* Oxford: Oxford University Press, 1986, 144–195.

Gödel, Kurt. 1934. "On Undecidable Propositions of Formal Mathematical Systems." In Feferman, Solomon (ed) *Kurt Gödel. Collected Works. Vol. 1.* Oxford: Oxford University Press, 1986, 346–369.

Gödel, Kurt. 193?. "[Undecidable Diophantine Propositions]." In Feferman, Solomon (ed) *Kurt Gödel. Collected Works. Vol. 3.* Oxford: Oxford University Press, 1995, 164–175.

Gödel, Kurt. 1944. "Russel's Mathematical Logic." In Feferman, Solomon (ed) *Kurt Gödel. Collected Works. Vol. 2.* Oxford: Oxford University Press, 1990, 119–141.

Gödel, Kurt. 1951. "Some Basic Theorems on the Foundations of Mathematics and their Implications." In Feferman, Solomon (ed) *Kurt Gödel. Collected Works. Vol. 3.* Oxford: Oxford University Press, 1995, 304–323.

Gödel, Kurt. 1964. "Postscript to Gödel's (1934)." In Feferman, Solomon (ed) *Kurt Gödel. Collected Works. Vol. 1.* Oxford: Oxford University Press, 1986, 369–371.

Gödel, Kurt. 1972. "On an Extension of Finitary Mathematics which has not yet been used." In Feferman, Solomon (ed) *Kurt Gödel. Collected Works. Vol. 2.* Oxford: Oxford University Press, 1990, 271–280.

Gödel, Kurt. 1972a. "Some Remarks on the Undecidability Results." In Feferman, Solomon (ed) *Kurt Gödel. Collected Works. Vol. 2.* Oxford: Oxford University Press, 1990, 305–306.

Griffiths, Thomas and Zoubin Ghahramani. 2005. "Infinite latent feature models and the Indian buffet process." Technical Report 2005-001, Gatsby Computational Neuroscience Unit.

Griffiths, Thomas and Zoubin Ghahramani. 2011. "The Indian Buffet Process: An Introduction and Review." *Journal of Machine Learning Research* 12: 1185–1224.

Gurevich, Yuri. 2003. "Abstract State Machines Capture Parallel Algorithms." *ACM Transactions on Computational Logic* 4, no. 4: 578–651.

Herbrand, Jacques. 1931. "On the Consistency of Arithmetic." In Goldfarb, Warren (ed) *Jacques Herbrand: Logical Writings.* Cambridge: Harvard University Press, 1971, 282–298.

Herbrand, Jacques. 2003. "Correspondence between Herbrand and Gödel." In Feferman, Solomon (ed) *Kurt Gödel. Collected Works. Vol. 5.* Oxford: Oxford University Press, 14–25.

Hilbert, David and Paul Bernays. 1939. *Grundlagen der Mathematik II.* Berlin: Springer Verlag. 2nd Edition, 1970.

Hodges, Andrew. 2005. "Can Quantum Computing Solve Classically Unsolvable Prob- lems?" quant-ph/0512248.

Juurlink, Ben and Harry Wijshoff. (1996). "The E-BSP Model: Incorporating General Locality and Unbalanced Communication Into the BSP Model" In Bougé, Luc, Pierre Fraigniaud, Anne Mignotte, and Yves Robert (eds) *Second International Euro-Par Conference Lyon, France, August 26–29, 1996 Proceedings, Volume II*: 339–347.

Kalmár, László. 1959. "An Argument Against the Plausibility of Church's Thesis." In Heyting, Arend (ed) *Constructivity in Mathematics, Proceedings of the Colloquium Held at Amsterdam.* Amsterdam: North–Holland Publishing Company, 72–80.

Kieu, Tien. 2002. "Quantum Hypercomputation." *Minds and Machines* 12, no. 4: 541–561.

Kleene, Stephen. 1936. "General Recursive Functions of Natural Numbers." *Mathematische Annalen* 112, no. 5: 727–742.

Kleene, Stephen. 1938. "On Notation for Ordinal Numbers." *The Journal of Symbolic Logic* 3, no. 4: 150–155.

Kleene, Stephen. 1944. "On the Form of the Predicates in the Theory of Constructive Ordinals." *The American Journal of Mathematics* 66, no. 1: 41–58.

Kleene, Stephen. 1952. *Introduction to Metamathematics.* Amsterdam: North–Holland Publishing Company.

Kleene, Stephen. 1981. "Origins of Recursive Function Theory." *Annals of the History of Computing*, Vol. 3, no 1: 52–67.

Kleene, Stephen. 1986. "Introduction to Gödel's (1934)." In Feferman, Solomon (ed) *Kurt Gödel. Collected Works. Vol. 1.* Oxford: Oxford University Press, 338–345

Kreisel, Georg. 1965. "Mathematical Logic." In Saaty, Thomas (ed) *Lectures on Modern Mathematics. Vol 3.* Wiley. 95–195.

Kreisel, Georg. 1967. "Mathematical Logic: What has it Done for the Philosophy of Mathematics?" In Shoemann, Ralph (ed) *Bertrand Russell. Philosopher of the Century.* Allen and Unwin. 201–272.

Kreisel, Georg. 1967b. "Informal Rigour and Completeness Proofs." In Lakatos, Imre (ed) *Problems in the Philosophy of Mathematics.* Amsterdam: North–Holland Publishing, 138–186.

Kreisel, Georg. 1969. "Foundations of Mathematics: 1900-1950." In Harré, Romano (ed) *Scientific Thought 1900-1960: A Selective Survey.* Oxford: Calderon Press. 4–13.

Kreisel, Georg. 1970. "Church's Thesis: A Kind of Reducibility Axiom for Constructive Mathematics." In Kino, Akiko, John Myhill and Richard Vesley (eds) *Intuitionism and Proof Theory. Proceedings of the Summer Conference, Buffalo. 1968.* Amsterdam: North–Holland Publishing. 121–150.

Kreisel, Georg. 1970b. "The Formalist-Positivist Doctrine of Mathematical Precision in the Light of Experience." *L'Áge de la Sciencec* 3, no. 1: 2–46.

Kreisel, Georg. 1971. "Some Reasons for Generalizing Recursion Theory." In Gandy Robin and Charles Yates (eds) *Logic Colloquium '69. Proceedings of the Summer school and Colloquium in Mathematical Logic, Manchester, August 1969.* Amsterdam: North-Holland Publishing. 139–198.

Kreisel, Georg. 1971b. "A Survey of Proof Theory II." In Fenstad, Jens Erik (ed) *Proceedings of the Second Scandinavian Logic Symposium.* Amsterdam: North-Holland Publishing. 109–170.

Kreisel, Georg. 1972. "Which Number Theoretic Problems can be Solved in Recursive Progressions on Π^1_1-paths Through O." *The Journal of Symbolic Logic* 37, no. 2: 311–334.

Lagnado, David and Steven Sloman. 2004. "The Advantage of Timely Intervention." *Journal of Experimental Psychology: Learning, Memory, and Cognition* 30, no. 4: 856–876.

Lagnado, David and Steven Sloman. 2006. "Time as a Guide to Cause." *Journal of Experimental Psychology: Learning, Memory, and Cognition* 32, no. 3: 451–460.

Lagnado, David, Michael Waldmann, York Hagmayer, and Steven Sloman. 2007. "Beyond Covariation: Cues to Causal Structure." In Gopnik, Alison and Laura Schultz (eds) *Causal Learning: Psychology, Philosophy, and Computation.* Oxford: Oxford University Press: 154–172.

Lamport, Leslie. 1979. "How to Make a Multiprocessor Computer that Correctly Executes Multiprocess Programs." *IEEE Transactions on Computers* c-28, no. 9: 690–691.

Lamport, Leslie and Nancy Lynch. 1990. "Distributed Computing: Models and Methods." In van Leeuwen, Jan (ed) Handbook of Theoretical Computer Science. Groningen: Elsevier, 1157–1199.

Leopold, Claudia. 2001. *Parallel and Distributed Computing.* New York: John Wiley and Sons Inc.

Love, Bradley, Douglas Medin and Todd Gureckis. 2004. "SUSTAIN: A Network Model of Category Learning." *Psychological Review* 111, no 2: 309–332.

Marsella, Stacy, Jonathan Gratch, and Paolo Petta. 2010. "Computational Models of Emotion." In Scherer, Klaus, Tanja Bänziger, and Etienne Roesch (eds) *A Blueprint for Affective Computing: A Sourcebook and Manual.* Oxford: Oxford University Press. 21–41.

Mendelson, Elliott. 1990. "Second Thoughts about Church's Thesis and Mathematical Proofs." *The Journal of Philosophy* 87, no. 5: 225–233.

Mendelson, Elliott. 2006. "On the Impossibility of Proving the 'Hard-half' of Church's Thesis." In Olszewski, Adam, Jan Wolenski and Robert Janusz (eds) *Church's Thesis After 70 Years.* Frankfurt: Ontos Verlag. 304–309.

Nielsen, Michael and Isaac Chuang. 2000. *Quantum Computation and Quantum Information.* Cambridge: Cambridge University Press.

Oatley, Keith and Philip Johnson-Laird. 2013. "Cognitive Approaches to Emotions." *Trends in Cognitive Sciences* 20: 1–7.

Pearl, Judea. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishing.

Piccinini, Gualtiero. 2017. "Computation in Physical Systems." Stanford Encyclopedia of Philosophy: https://plato.stanford.edu/entries/computation-physicalsystems/ Accessed online August 1st, 2017.

Post, Emil. 1936. "Finite Combinatory Processes – Formulation 1." *The Journal of Symbolic Logic* 1, no. 3: 103–105.

Post, Emil. 1944. "Recursively Enumerable Sets of Positive Integers and Their Decision Problems." *American Mathematical Society* 50, no. 5: 284–316.

Post, Emil. 1965. "Absolutely Unsolvable Problems and Relatively Undecidable Propositions – Account of an Anticipation." In Davis, Martin (ed) *The Undecidable.* Mineola: Dover Publications. 2004. 340–406.

Pour-El, Marian and Jonathan Richards. 1989. *Computability in Analysis and Physics.* Berlin: Springer Verlag.

Prinz, Jesse. 2004. *Gut Reactions.* Oxford: Oxford University Press.

Rathjen, Michael and Wilfried Sieg. (2015). "Proof Theory." Draft.

Rieffel, Eleanor and Wolfgang Polak. 2011. *Quantum Computing. A Gentle Introduction.* Cambridge: MIT Press.

Shapiro, Stewart. 1981. "Understanding Church's Thesis." *The Journal of Philosophical Logic* 10, no. 3: 353–361.

Shapiro, Stewart. 1993. "Understanding Church's Thesis, Again." *Acta Analytica* 11: 59–77.

Shapiro, Stewart. 2006. "Computability, Proof, and Open-texture." In Olszewski, Adam, Jan Wolenski and Robert Janusz (eds) *Church's Thesis After 70 Years.* Frankfurt: Ontos Verlag. 420–455.

Shapiro, Stewart. 2015. "Proving Things About the Informal." In Sommaruge, Thomas and Thomas Strahm (eds) *Turing's Revolution. The Impact of His Ideas about Computability.* Basel: Birkhäuser. 283–296.

Shoenfield, Joseph. 1991. *Recursion Theory.* Berlin: Springer Verlag.

Sieg, Wilfried. 1994. "Mechanical Procedures and Mathematical Experience." In George, Alexander (ed) *Mathematics and Mind.* Oxford: Oxford University Press, 71–117.

Sieg, Wilfried. 1997. "Step by Recursive Step: Church's Analysis of Effective Calculability." *The Bulletin of Symbolic Logic* 3, no. 2: 154–180.

Sieg, Wilfried. 2002. "Calculations by Man and Machine: Mathematical Presentation." In Gärdenfors, Peter, Jan Woleński and Katarzyna Kijania-Placek (eds) *Proceedings of the Cracow International Congress of Logic, Methodology and Philosophy of Science Vol I*, Synthese Series, Dordrecht: Kluwer Academic Publishers, 247–262.

Sieg, Wilfried. 2003. "Introduction to the Correspondence between Herbrand and Gödel." In Feferman, Solomon (ed) *Kurt Gödel. Collected Works. Vol. 5.* Oxford: Oxford University Press, 3–13.

Sieg, Wilfried. 2005. "Only Two Letters: The correspondence between Herbrand and Gödel." *The Bulletin of Symbolic Logic* 11, no. 2: 172–184.

Sieg, Wilfried. 2008. "Church without Dogma – Axioms for Computability." In Löwe, Sorbi, and Cooper (eds) *New Computational Paradigms.* Springer Verlag. 139–152.

Sieg, Wilfried. 2009. "On Computability." In Irvine, Andrew (ed) *Philosophy of Mathematics (Handbook of the Philosophy of Science).* Amsterdam: North–Holland Publishing Company, 535–630.

Sieg, Wilfried. 2012. "Axioms for Computability: Do They Allow a Proof of Church's Thesis?" In Zenil, Hector (ed) *A Computable Universe Understanding and Exploring Nature as Computation.* New Jersey: World Scientific. 99–123.

Sieg, Wilfried. 2013. "Gödel's Philosophical Challenge (to Turing)." In Copeland, Jack, Carl Posy and Oron Shagrir (eds) *Computability: Turing, Gödel, Church, and Beyond.* Cambridge: MIT Press. 183–202.

Sieg, Wilfried and John Byrnes. 1999. "An Abstract Model for Parallel Computations: Gandy's Thesis." *The Monist* 82, no. 1: 150–164.

Sieg, Wilfried, Máté Szabó and Dawn McLaughlin. 2017. "Why Post did [not] have Turing's Thesis." Omodeo and Policriti (eds): *Martin Davis on Computability, Computational Logic, and Mathematical Foundations.* Springer International Publishing, 175–208.

Siegelmann, Hava. 1999. *Neural Networks and Analog Computation. Beyond the Turing Limit.* Boston: Birkhäuser.

Skillicorn, David, Jonathan Hill and Bill McColl. 1996. *Questions and Answers About BSP.* Technical Report PRG-TR-15-96, Programming Research Group, Oxford University Computing Laboratory.

Skolem, Thoralf. 1923. "The Foundations of Elementary Arithmetic Established by Means of the Recursive Mode of Thought, without the Use of Apparent Variables Ranging Over Infinite Domains." In van Heijenoort, Jean (ed) *From Frege to Gödel. A Source Book in Mathematical Logic. 1879-1931.* Cambridge: Harvard University Press. 1967. 302–333.

Spirtes, Peter, Clark Glymour and Richard Scheines. 1993/200. *Causation, Prediction and Search.* Cambridge, Massachusetts: The MIT Press.

Steyvers, Mark, Joshua Tenenbaum, Eric-Jan Wagenmakers and Ben Blum. 2003. "Inferring Causal Networks from Observations and Interventions." *Cognitive Science* 27: 453–489.

Turing, Alan. 1936. "On Computable Numbers, With an Application to the Entscheidungsproblem." *Proceedings of the London Mathematical Society*, Series 2, Vol. 42: 230–265.

Turing, Alan. 1938. "Systems of Logic Based on Ordinals. PhD Dissertation." In Davis, Martin (ed) *The Undecidable*. Mineola: Dover Publications. 2004. 340–406.

Turing, Alan. 1950. "The Word Problem in Semi-Groups with Cancellation." *The Annals of Mathematics*, Series 2, Vol. 52, no. 2: 491–505.

Turing, Alan. 1954. "Solvable and Unsolvable Problems." *Science News* 31, 7–23.

Uspensky, Victor. 1987. *Gödel's Incompleteness Theorem.* Moscow: Mir Publishers.

Valiant, Leslie. 1990. "A Bridging Model for Parallel Computation." *Communications of the ACM* 33, no. 8: 103–111.

Valiant, Leslie. 1994. *Circuits of the Mind.* New York: Oxford University Press.

Valiant, Leslie. 2006. "A Quantitative Theory of Neural Computation." *Biological Cybernetics* 95, no. 3: 205–211.

van Heijenoort, Jean. 2003. "Correspondence between van Heijenoort and Gödel." In Feferman, Solomon (ed) *Kurt Gödel. Collected Works. Vol. 5.* Oxford: Oxford University Press, 307–325.

Velásquez, Juan. 1997. "Modeling Emotions and Other Motivations in Synthetics Agents." In *Proceedings of the 1997 National Conference on Artificial Intelligence* (AAAI97). Providence, RI. 10–15.

von Neumann, John. 1966. *Theory of Self-Reproducing Automata*. Edited and Completed by Arthur Burks. Urbana: University of Illinois Press.

Waldmann, Michael, Laura Martignon. 1998. "A Bayesian Network Model of Causal Learning." In Gernsbacher, Morton Ann and Sharon Derry (eds) *Proceedings of the Twentieth Annual Meeting of the Cognitive Science Society, University of Wisconsin, Madison*, 1998. Psychology Press: 1102–1107.

Wang, Hao. 1987. *Reflections on Kurt Gödel*. Cambridge: MIT Press.

Wellen, Sarah, David Danks. 2012. "Learning Causal Structure Through Local Prediction-Error Learning." In Miyake, Naomi, David Peebles and Richard Cooper (eds) *Proceedings of the 34th annual conference of the cognitive science society, Austin, TX*. Cognitive Science Society: 2529–2534.